1990年1月1日発行(毎月1回1日発行)第9巻1号通巻93号 昭和58年11月2日第三種郵便物認可

ISSN 0910-7614

パーソナルコンピュータ・マガジン
MZシリーズ,X1/turbo,X68000&ポケコン

Oh!X

オー!エックス 定価560円

特集1
オペレーティングスタイルの研究
特集2
Cプログラミング応用編

1989年度GAME OF THE YEAR
ノミネート作品発表

特別付録X68000ゲームソフトカタログ

X1/turbo
シミュレーションゲームSuper Battle

X68000
DōGA・CGアニメーション講座
C調言語講座PRO-68K/マシン語プログラミング

S-OS
SLANG用アクションゲームWORM KUN
再掲載・構造型コンパイラ言語SLANG

THE SOFTOUCH
レナム/メタルサイト

LIVE in '90
X1/turboさよならを過ぎて
X68000RYDEEN

猫とコンピュータ/知能機械概論
マシン語カクテル/ショートプロぱーてい

1
JAN. 1990

夢のつづきを語ろう。

アーティスティックな側面にばかり気をとられていると、X68000の本質を見失ってしまうかも知れません。X68000が、もとよりホリゾンタルなマシンとしての不偏性を有していたことについて、異論をはさむ余地はないでしょう。あれだけ先鋭な仕事をこなしてきたこのマシンに普通の仕事がこなせないわけはないからです。いわば68000の潜在能力でしょうか。このCPUを決断した私たちは、当然「今」もそれにつづく未来をも照準に入れていました。とことん活かしきるには時間が必要です。そして、それがまた本当のユーザーインターフェイスとして結ばれてくるのです。汎用性といえばいささか平凡ですが、まさに真の意味での汎用性を謳えるマシンはそう多くないはずです。これまで圧倒的なご支持をいただいた感性豊かなユーザー、ソフトハウス、パブリッシャー、ハードベンダー各位の熱い視線が、ここにきてまた、X68000のソフト/ハード環境に新たな局面をひらこうとしています。

〈共通特長〉●さらに高い次元へと進化した処理機能とヒューマンインターフェイス、Human 68 k ver 2. 0、日本語フロントエンドプロセッサver2. 0搭載●プロセッサの未来を先取りした68000搭載●テキスト、グラフィック、スプライトの3画面を独立させた独自のメモリアーキテクチャー●1024×1024ドット(最大表示エリア768×512ドット)、高品位な金属までも自然に表現しうる65,536色同時発色(512×512ドット時)の高解像度自然色グラフィックス●16×16ドットの緻密なキャラクタを駆使できるスプライト機能(水平32スプライト、1画面128スプライト、65,536色中16色)●リアルなサウンドシーンをクリエイトできるステレオFM音源に加え、サンプリング音源としてAD PCM搭載●オートロード、オートイジェクトメカ採用、インテリジェントな1Mバイトの5″FDD2基搭載●蓄積された多彩なジャンルのアプリケーションが利用できるX68000シリーズとソフトコンパチ。

〈EXPERTシリーズ〉●高密度実装を象徴するフォルム、マンハッタンシェイプ●新たな領域をひらく3Mバイトの大容量メモリを標準装備、メインメモリは標準で2Mバイト、最大12Mバイトまで拡張可能●40Mバイトハードディスク搭載(CZ-612C)※●マウス・トラックボール標準装備●日本語入力にスムーズに対応するASCII準拠フルキーボードを採用。

〈PROシリーズ〉●意表をつくボディコンストラクション、高度な実装技術に裏付けられた洗練と信頼性の新しいスタンダードフォルム●高度なシステム化への対応を考慮した4スロットの拡張I/0スロット標準装備●プロニーズに対応した大容量ファイル、40Mバイトハードディスク搭載(CZ-662C)※●2Mバイトの大容量メモリを標準装備●マウス標準装備●使いやすいワイドスケールのフルキーボード。

※CZ-602C、CZ-652Cには、本体内に内蔵できる増設用の40Mバイトハードディスクドライブ(CZ-64H標準価格120,000円税別・取付費別)をサポート。

EXPERT・PRO

選べる4タイプのディスプレイをサポート

15型カラーディスプレイテレビ	(ドットピッチ0. 39mm)	CZ-602D-GY(グレー)・-BK(ブラック)	標準価格 99,800円(チルトスタンド同梱・税別)
15型カラーディスプレイテレビ	(ドットピッチ0. 31mm)	CZ-612D-GY(グレー)・-BK(ブラック)	標準価格119,800円(チルトスタンド同梱・税別)
14型カラーディスプレイ	(ドットピッチ0. 31mm)	CZ-603D-GY(グレー)・-BK(ブラック)	標準価格 84,800円(チルトスタンド同梱・税別)
14型カラーディスプレイ	(ドットピッチ0. 31mm)	CZ-604D-GY(グレー)・-BK(ブラック)	90年1月発売予定(スピーカー2個/チルトスタンド同梱)

EXEリーダーズグッズ
プレゼント実施中

●いま、EXE会員よりご紹介のお客様がEXEショップでX68000シリーズを購入されますと、EXE会員にEXEリーダーズグッズをプレゼントします。詳しくはEXEショップにお問い合わせください。
●また、X68000シリーズをご購入のお客様は、ぜひEXEクラブにご入会ください。

本広告に掲載しております商品および役務の価格には消費税は含まれておりませんので、ご購入の際、消費税額をお支払い下さい。

シャープ株式会社

●お問い合わせは…シャープ㈱電子機器事業本部システム機器営業部〒545大阪市阿倍野区長池町22番22号☎(06)621-1221(大代表)
電子機器事業本部テレビ事業部第4商品企画部〒162東京都新宿区市谷八幡町8番地☎(03)260-1161(大代表)

表紙絵： Moto Noriyuki

■広告目次

Oh!X

CONT

〈スタッフ〉
●編集長／前田　徹　●編集／植木章夫　太田慎一　岡崎栄子　●協力／有田隆也　中森　章　後藤貴行　林　一樹　荻窪　圭　岡本浩一郎　毛内俊行　吉田賢司　影山裕昭　相馬英智　古村　聡　村田敏幸　丹　明彦　三沢和彦　長沢淳博　宮島　靖　金子俊一　●カメラ／杉山和美　●イラスト／永沢しげる　山田晴久　小栗由香　●アートディレクター／島村勝頼　●レイアウト／元木昌子　AD GREEN　●校正／千野延明　織田洋子

1990 JAN. 1

E N T S

特集2 再帰大作戦

DōGA・CGA

Super Battle

レナム

メタルサイト

Xstor

SHARP

クリエイティブマインドあふれる周辺機器が

CZ-600C/601C/611C/602C/612C

ディスプレイ関連

カラーディスプレイテレビ

15型カラーディスプレイテレビ
CZ-602D-GY・-BK
標準価格 99,800円(税別)
(チルトスタンド同梱)

15型カラーディスプレイテレビ
CZ-612D-GY・-BK
標準価格 119,800円(税別)
(チルトスタンド同梱)

カラーディスプレイ

NEW

21型カラーディスプレイ
CU-21HD
標準価格 148,000円(税別)
(スピーカー2個同梱)

NEW

14型カラーディスプレイ
CZ-604D-GY・-BK
90年1月発売予定
(スピーカー2個・チルトスタンド同梱)

14型カラーディスプレイ
CZ-603D-GY・-BK
標準価格 84,800円(税別)
(チルトスタンド同梱)

チューナー

RGBシステムチューナー
CZ-6TU-GY・-BK
標準価格 33,100円(税別)
(リモコン付)

CRTフィルター

高性能CRTフィルター
BF-68PRO
標準価格 19,800円(税別)
(14/15型用)

アートツール

画像入力

カラーイメージスキャナ※1
CZ-8NS1
標準価格 188,000円(税別)

スキャナ用パラレルボード
CZ-6BN1
標準価格 29,800円(税別)

映像入力

カラーイメージユニット
CZ-6VT1
CZ-6VT1-BK
標準価格 69,800円(税別)

プリンタ

カラープリンタ

24ドット
熱転写カラー漢字プリンタ
CZ-8PC3
標準価格 65,800円(税別)
(信号ケーブル同梱)

48ドット
熱転写カラー漢字プリンタ
CZ-8PC4
CZ-8PC4-GY
標準価格 99,800円(税別)
(信号ケーブル同梱)

カラービデオプリンタ

カラービデオプリンタ
★CZ-6PV1
標準価格 198,000円(税別)
(信号ケーブル同梱)

カラーイメージジェット

NEW

カラーイメージジェット※2
IO-735X
標準価格248,000円(税別)
(信号ケーブル別売)

ドットプリンタ

NEW

24ピンカラー漢字プリンタ(80桁)
CZ-8PG1
標準価格 130,000円(税別)
(信号ケーブル同梱)

NEW

24ピンカラー漢字プリンタ(136桁)
CZ-8PG2
標準価格 160,000円(税別)
(信号ケーブル同梱)

NEW

24ピン漢字プリンタ(136桁)
CZ-8PK10
標準価格 97,800円(税別)
(信号ケーブル同梱)

ファイル

ハードディスク

ハードディスクユニット(20MB)
CZ-620H
標準価格 178,000円(税別)

増設用ハードディスクドライブ
(40MB)
CZ-64H
標準価格 120,000円(税別)
(取付費別)

※取付に関してはシャープお客様ご相談窓口にてご相談ください。

※1 ご使用に際しては、カラーイメージスキャナCZ-8NS1に同梱のRS-232Cケーブルで接続するか、より高速のパラレルデータ伝送を行う場合、別売のスキャナ用パラレルボードCZ-6BN1標準価格29,800円(税別)で接続してください。
※2 別売の信号ケーブルIO-73CX標準価格5,500円(税別)で接続して下さい。
※3 CZ-652C、662Cをお持ちの方は包装箱の表示形名 CZ-6BE 1Aの右横にⒶマーク表示のあるものをお買い求めください。

X1・X1turbo シリーズ用 周辺機器

標準価格は税別です。

カラーディスプレイ		
●21型カラーディスプレイ※1	CU-21HD	148,000円

映像・画像入力編集装置		
●カラーイメージスキャナ	CZ-8NS1	188,000円
●カラーイメージボードII	CZ-8BV2	39,800円

●立体映像セット	★CZ-8BR1	29,800円
●パーソナルテロッパ※2	CZ-8DT2	44,800円

FM音源		
●ステレオタイプFM音源ボード	CZ-8BS1	23,800円

スピーカー(2本1組)標準装備、ミュージックツール同梱

プリンタ		
●24ピンカラー漢字プリンタ(80桁)	CZ-8PG1	130,000円
●24ピンカラー漢字プリンタ(136桁)	CZ-8PG2	160,000円
●24ピン漢字プリンタ(136桁)	CZ-8PK10	97,800円
●24ドット熱転写カラー漢字プリンタ	CZ-8PC3	65,800円
●48ドット熱転写カラー漢字プリンタ	CZ-8PC4	99,800円
●48ドット熱転写カラー漢字プリンタ	CZ-8PC4-GY	99,800円
●カラービデオプリンタ	★CZ-6PV1	198,000円
●カラーイメージジェット	IO-735X	248,000円

ファイル		
●ミニフロッピーディスクユニット(2HD・2D)※3	★CZ-520F	118,000円

シャープ株式会社 ●お問い合わせは…シャープ㈱電子機器事業本部システム機器営業部 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)

資料請求券 X68000ペリフェラル oh!X 1体

X68000をサポート。

シャープペリフェラルファミリー X68000

CZ-652C/662C

ボード

拡張メモリ

1MB増設RAMボード
(CZ-600C用)
CZ-6BE1
標準価格 35,000円(税別)

1MB増設RAMボード※3
(CZ-601C/611C/652C/662C用)
CZ-6BE1A
標準価格 38,000円(税別)

2MB増設RAMボード※4
CZ-6BE2
標準価格 79,800円(税別)

4MB増設RAMボード※4
CZ-6BE4
標準価格 138,000円(税別)

LANボード

LANボード
CZ-6BL1
標準価格 268,000円(税別)
※電源ユニット・ソフトウェア
(ネットワークドライバVer1.0)同梱

インターフェイス

ユニバーサルI/Oボード
CZ-6BU1
標準価格 39,800円(税別)

GP-IBボード
CZ-6BG1
標準価格 59,800円(税別)

増設用RS-232Cボード
(2チャンネル)
CZ-6BF1
標準価格 49,800円(税別)

数値演算プロセッサ

数値演算プロセッサボード
CZ-6BP1
標準価格 79,800円(税別)

FAX

FAXボード
CZ-6BC1
標準価格 79,800円(税別)

MIDI

MIDIボード
CZ-6BM1
標準価格 26,800円(税別)

ネットワーク

モデム

モデムユニット※5
CZ-8TM2
標準価格 49,800円(税別)
(RS-232Cケーブル同梱)

RS-232Cケーブル

RS-232Cケーブル
(平行接続型)
CZ-8LM1
標準価格 7,200円(税別)

RS-232Cケーブル
(クロス接続型)
CZ-8LM2
標準価格 7,200円(税別)

入力

インテリジェントコントローラ
CZ-8NJ2
標準価格 23,800円(税別)

マウス・トラックボール
CZ-8NM3
標準価格 9,800円(税別)

トラックボール
CZ-8NT1
標準価格 13,800円(税別)

マウス
CZ-8NM2A
標準価格 6,800円(税別)

ジョイカード
CZ-8NJ1
標準価格 1,700円(税別)

その他

拡張スロット

拡張I/Oボックス(4スロット)
(CZ-600C/601C/611C/602C/612C用)
CZ-6EB1
CZ-6EB1-BK
標準価格 88,000円(税別)

スピーカー

アンプ内蔵
スピーカーシステム(2本1組)
AN-S100
標準価格 36,600円(税別)

システムラック

システムラック
(CZ-600C/601C/611C/602C/612C用)
CZ-6SD1
標準価格 44,800円(税別)

※4 ご使用に際しては、あらかじめ別売の1MB増設RAMボードCZ-6BE1 標準価格35,000円(税別・CZ-600C用)、CZ-6BE1A 標準価格38,000円(税別・CZ-601C、CZ-611C、652C、662C用)を増設してください。
※5 モデムユニットCZ-8TM2に同梱のソフトはX1/X1ターボシリーズ用です。

品名	品番	価格
●ミニフロッピーディスクユニット(2D)	★CZ-502F	99,800円
●ミニフロッピーディスクユニット(2D・1ドライブ)	CZ-503F	49,800円
●増設用ミニフロッピーディスクドライブ(2D)※4	CZ-53F-BK	19,800円

拡張ボード・その他

品名	品番	価格
●モデムユニット(300/1200ボー)	CZ-8TM2	49,800円
●320KB外部メモリ	CZ-8BE2	29,800円
●RS-232C・マウスボード※5	CZ-8BM2	19,800円
●フロッピーディスクインターフェイス※6	CZ-8BF1	14,800円
●JIS第1水準漢字ROM※7	CZ-8BK2	19,800円
●RS-232C用ケーブル(平行接続型)	CZ-8LM1	7,200円
●RS-232C用ケーブル(クロス接続型)	CZ-8LM2	7,200円
●拡張I/Oボックス	CZ-8EB3	33,800円
●RFコンバータ※8	AN-58C	2,980円
●インテリジェントコントローラ	CZ-8NJ2	23,800円
●マウス・トラックボール	CZ-8NM3	9,800円
●マウス	CZ-8NM2A	6,800円
●トラックボール	CZ-8NT1	13,800円
●ジョイカード	CZ-8NJ1	1,700円
●チルトスタンド※9	CZ-6ST1-E・-B	5,800円
●高性能CRTフィルター※10	BF-68PRO	19,800円
●スキャナ用パラレルボード※11	CZ-8BN1	27,800円

●品番中の-表示は、B〈ブラック〉・E〈オフィスグレー〉を示します。※1 X1ターボZシリーズ用 ※2 CZ-862Cには接続できません ※3 X1ターボシリーズ用 ※4 CZ-830C用 ※5 X1シリーズ用 ※6 CZ-850CでCZ-520Fを使用する場合に必要 ※7 CZ-800C、801C、802C、803C、811C、820C用 ※8 CZ-820C、822C、830C用 ※9 CZ-600D、880D、830D用 ※10 14/15型用 ※11 CZ-8NS1用 ●接続等の説明につきましては、周辺機器総合カタログをご参照ください。

★印の商品は在庫僅少です。

本広告に掲載しております商品および役務の価格には消費税は含まれておりませんので、ご購入の際、消費税額をお支払い下さい。

電子機器事業本部テレビ事業部第4商品企画部 〒162 東京都新宿区市谷八幡町8番地 ☎(03)260-1161(大代表)

SHARP

"アート"と呼べる高水準のソフトウェアが

データと上手につきあう法、教えます。情報人の24時間をマネージメント、「サイバーノート」新登場。

プライベートなデータやビジネスデータを簡単な操作で管理・運営できるパーソナルデータベースです。リフィル、タックシール、ハガキなどへの印字もOK。シャープ電子手帳とのデータ変換（別売の通信ケーブルが必要）も実現。電子手帳をX68000の情報端末として利用できます。

●**住所録/名刺管理/電話帳総合管理機能**：最大32760件/1ファイルの大容量データ管理。名刺管理では画像データの表示も可能。●**カレンダー機能**●**スケジュール機能**●**家計簿管理機能**●**メモ管理機能**●**高速マルチ検索機能**●世界時計/時計/バイオリズム/電卓など多彩な**アクセサリー機能**●**各種出力フォーム**を装備：システム手帳リフィル（バイブルサイズ）、A4、A5、連続帳票、宛名ラベル、ハガキなどに対応●ファイル形式は「CARD PRO-68K」と完全コンパチブル。

CYBERNOTE PRO-68K

CZ-243BS 標準価格19,800円（税別）

必要なとき、いつでも使える、サッと呼び出せる。メモリ常駐型のステーショナリーソフトウェア。

他のソフトを実行中でも呼び出して使える便利ツール。使い方は簡単、他のアプリケーションを起動する前に、このソフトを一度起動するだけ。これで、他のアプリケーション実行中にも、「メモ」や「スケジュール」、「住所録」などStationery PRO-68Kの持つ多彩な機能がワンタッチで使えます。また、X68000上で入力したデータをシャープ電子手帳の「電話帳」、「スケジュール」、「メモ」へ送信したり、逆に電子手帳側からデータを受信して編集することができます（別売の通信ケーブルが必要）。

Stationery PRO-68K

CZ-240BS 標準価格14,800円（税別）

シャープ株式会社　●お問い合わせは…シャープ㈱電子機器事業本部テレビ事業部 第4商品企画部 〒162 東京都新宿区市谷八幡町8番地 ☎(03)260-1161（大代表）へ。

資料請求券 CZソフト oh!X 1条

X68000をサポート。

シャープオリジナルソフトウェア
X68000

サウンドツール

Musicstudio PRO-68K ver.1.1
■CZ-252MS 標準価格28,800円(税別)
24トラック対応MIDIマルチレコーディングソフトMusicstudio PRO-68Kがバージョンアップしました。従来の機能に加え、小節間のコピー及びデリートや、MIDIインプットモニターなど、数々の機能を追加・改良。さらに使いやすくなりました。
※MIDIボード(CZ-6BM1)が必要です。

MUSIC PRO-68K〔MIDI〕
■CZ-247MS 標準価格28,800円(税別)
MIDI対応自動伴奏機能をサポート、簡単な楽譜入力で演奏が楽しめます。
※MIDIボード(CZ-6BM1)が必要です。

ソングライブラリ〈101曲集〉
■CZ-248MS 標準価格8,800円(税別)
鑑賞用と音楽データ加工作成用からなるライブラリです。

Sampling PRO-68K
■CZ-215MS 標準価格17,800円(税別)
AD PCM機能を活かす高機能サンプリングエディタ。多彩なEDITORを装備、サンプリング音のデータはBASICでも活用できます。

SOUND PRO-68K
■CZ-214MS 標準価格15,800円(税別)
スタジオのコンソールパネルを操作する感覚でFM音源による音創りが楽しめるサウンドエディタ。

MUSIC PRO-68K
■CZ-213MS 標準価格18,800円(税別)
最大8パートのスコア(総譜)が書け、内蔵のFM音源で演奏できる楽譜ワープロ＆演奏用ツール。

アートツール

NEW PrintShop PRO-68K
■CZ-221HS 標準価格19,800円(税別)
オリジナリティあふれるはがき等、簡単に作成、印刷できるホームプロダクティビリティツール。ほとんどの処理をアイコンで表示しマウスで選ぶフレンドリーオペレーション。

グラフィックライブラリ VOL.1
■CZ-235GS 標準価格8,800円(税別)
暑中見舞用を中心としたNEW Print Shop PRO-68K用グラフィックデータ集。

グラフィックライブラリ VOL.2
■CZ-236GS 標準価格8,800円(税別)
年賀状を中心としたNEW Print Shop PRO-68K用グラフィックデータ集。

ビジネスツール

TOP給与計算エキスパート
■CZ-228BS 標準価格200,000円(税別)
給与計算から明細発行までを、リアルイメージ入力により自動的に、素早く処理することができます。

TOP財務会計
■CZ-227BS 標準価格200,000円(税別)
会計エキスパートシステムとデータベースを搭載し、機能と操作性を両立させた財務会計ソフト。

CARD PRO-68K
■CZ-226BS 標準価格29,800円(税別)
自由なレイアウト画面で入力できるワープロ機能を装備したカード型リレーショナルデータベース。

CARD PRO-68K用システム手帳リフィル集
■CZ-241BS 標準価格9,800円(税別)

CARD PRO-68K用活用フォーム集
■CZ-242BS 標準価格9,800円(税別)

DATA PRO-68K
■CZ-220BS 標準価格58,000円(税別)
コマンド入力の手間を軽減するヒストリー機能、罫線ドライバー付レポートライター機能、10進31桁の高精度演算。さらにイメージ表示機能を装備したコマンド型リレーショナルデータベースです。

BUSINESS PRO-68K
■CZ-212BS 標準価格68,000円(税別)
スプレッドシート(表計算)、データベース、グラフ作成機能を緊密に一体化させた統合ビジネスツールです。マウス対応のやさしいオペレーション、高度なエディタ機能、豊富な関数群など、初心者からプロまで幅広く使えます。

ゲーム

シューティングゲーム
〈ツインビー〉
■CZ-217AS
標準価格7,800円(税別)
©KONAMI 1988

シューティングゲーム
〈沙羅曼蛇〉
■CZ-218AS
標準価格8,800円(税別)
©KONAMI 1989

ブロックゲーム
〈アルカノイド〉
■CZ-222AS
標準価格7,800円(税別)
©TAITO CORP. 1987

ドライブゲーム
〈フルスロットル〉
■CZ-231AS
標準価格8,800円(税別)
©TAITO CORP. 1988

スポーツゲーム
〈熱血高校ドッジボール部〉
■CZ-232AS
標準価格7,800円(税別)
©TECHNOS JAPAN CORP. 1988

アクションゲーム
〈パックマニア〉
■CZ-233AS
標準価格7,800円(税別)
©NAMCO

アクションゲーム
〈ニュージーランドストーリー〉
■CZ-230AS
標準価格8,800円(税別)
©TAITO CORP. 1989

スポーツゲーム
〈V'BALL〉
■CZ-246AS
標準価格7,900円(税別)
©TECHNOS JAPAN CORP. 1989

通信ツール

Communication PRO-68K
■CZ-223CS 標準価格19,800円(税別)
300～19,200BPSまでの通信速度に対応し、各種データベースの漢字端末やパソコン通信に利用できます。逆スクロール機能、自動実行機能、コンカレント機能も装備。さらに豊富な編集機能をもった高機能通信ソフトです。

開発ツール

OS-9/X68000
■CZ-219SS 標準価格29,800円(税別)
X68000のもつグラフィック環境はもちろん、AD PCM音声、FM音源とグラフィックの同時再生といったマルチメディア機能をサポート。OS-9のもつマルチタスク機能、リアルタイム機能を活かした使い易く機能的なOS環境を提供します。また、これまでのデータ資産も活かせます。 ※OS-9はマイクロウェア社の登録商標です。

Human68k ver2.0
■CZ-244SS 標準価格9,800円(税別)
システムパフォーマンスを高める処理機能を付加したHuman 68kの最新バージョンです。マルチタスクに近い処理環境を提供するバックグラウンド処理、ネットワーク処理、ファイルアクセスのスピードアップなど、さらに高い次元へと進化した機能とユーザーインターフェイス。大容量メディアにも対応。

C compiler PRO-68K
■CZ-211LS 標準価格39,800円(税別)
Cコンパイラ、BASIC-Cコンバータ、アセンブラ、リンカ、デバッガ、アーカイバ、コンバータからなるツール。OS上のプログラム開発を効率良くサポートします。XCはC言語の基本的な仕様に準拠し、ANSI仕様も採用、ハードウェアをサポートした豊富なライブラリ(約700種)も用意されています。

THE福袋V2.0
■CZ-224LS 標準価格9,980円(税別)
アセンブラ、リンカ、デバッガ、アーカイバ、X-BASIC V2.00からなる手軽な開発ツールです。

AI-68K(Staff LISP/OPS PRO-68K)
■CZ-234LS 標準価格188,000円(税別)
AI開発用言語とエキスパート構築ツールがセットになったAIプログラム開発ツールです。

本広告に掲載しております商品および役務の価格には消費税は含まれておりませんので、ご購入の際、消費税額をお支払い下さい。

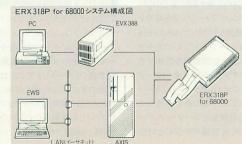
インサーキット・エミュレータ

ICE for X68000

16MHzで快調に動く、ザックスのERX318P for 68000。X68000用ゲームソフト開発に最適です。

クリエイティブワークステーション X68000を、さらにエキサイトさせる開発者へのザックスからの一つの解答。それがインサーキット・エミュレータ、ERX318P for 68000です。共通ユニットをEVX388とすると、PC-9801を中心とするMS-DOSマシーンに、またAXISを用いるとEWS(UNIXマシーン)をホストコンピュータとして使用することができます。CGやコンピュータ・ミュージックなど、ゲームソフトの開発には最適なICEです。

●システム価格:EVX388+ERX68000　¥1,624,000
（消費税別）　AXIS+ERX68000　¥2,224,000

ERX318P for 68000システム構成図

ERX318P for 68000の特長

- ●オブジェクトのロード時間:64Kバイトを27秒でロード
- ●ブレーク数:64K×4ポイント(ワイルドカード使用可能)
- ●ホストコンピュータ:PC-9801を中心とするMS-DOSマシン(EVX388システム)、EWSを中心としたLAN環境(AXISシステム)

Zax Corporation

株式会社 ザックス

本　　社/〒167 東京都杉並区荻窪5-20-12
TEL.03(392)3331(代) FAX.03(393)3878

大阪営業所/〒532 大阪市淀川区木川東3-5-21 第3丸善ビル
TEL.06(303)2671(代) FAX.06(303)2454

お問合わせは フリーダイヤル 0120-378-388 (ICD EVX)

感動の大容量。

Xstor 40 はシャープ X 68000専用に開発したハードディスクです。従来の汎用サブシステムにはない数々の特徴とハイセンスなデザインを実現した省スペースタイプの高品質ハードディスクです。

主な特徴

- ■厚さ35㎜。X68000本体の下にそのまま設置可能。
- ■平均アクセスタイム23ms。満足のいく高速性能を提供。
- ■パーソナルには余裕の40Mバイトの記憶容量。更に増設用HXD042を付加することにより最大80Mバイトまでのディスクシステムが利用可能。
- ■目的に応じた2モデルを用意。ハードディスクを初めて使う場合の1台目用と、すでにハードディスクを利用していて増設する場合の増設タイプを用意。
- ■Human 68K（Ver1.00以上）対応。既存の多くのソフトウェアがそのまま利用可能。
- ■交替セクタをユーザ領域から独立。しかも Formatプログラムにより自動実行。
- ■切電時のオートパーキングロックを採用。不意な衝撃に対しても磁気面を保護。
- ■高品質、低価格を実現。

HXD040：40MB/23ms/1台目用……………………………￥118,000
（X68000/ACE/EXPERT/PRO対応）

HXD042：40MB/23ms/2台目用……………………………￥128,000
（X68000/ ACE〈HD〉/EXPERT〈HD〉/PRO〈HD〉/HXD 040の増設用）

●データ転送速度/1.5MB/S ●増設/HXD042を1台増設可能 ●インターフェイス/SCSI（シングルユーザ）●交替処理/FORMATコマンドによるセクタ単位の自動交替処理 ●電源/入力AC100V 50/60Hz 消費電力25W（MAX）●外形寸法/35H×155W×313Dmm（突起物は含まず）●重量/約2.5kg

※SCSIのIDはHXD040は0番。HXD042は2番に固定されています。

SHARP X68000専用 ハードディスク

新・発・売

Xstor 40

《付属品》
接続ケーブル、取扱いマニュアル、メンテナンス登録カード、ターミネーター（HXD042のみ）

株式会社 アイテム
本社/〒251 神奈川県藤沢市南藤沢8-1-202 TEL.0466-27-1668 FAX.0466-27-2800
東京ショールーム/〒105 東京都港区新橋4-31-7 中村ビル7F TEL.03-434-4171 FAX.03-5472-5315

初めまして。私、専用機です。

充実した新機能

1. ハードディスク内蔵タイプにも接続することができます。
2. ID番号切り替えロータリースイッチにより1台〜最大8台まで接続が可能です。(Human 68K Ver2.0以上使用時)
3. ニューデザインで色もX68000シリーズ本体に合わせてブラック・グレーの2色を用意しました。
4. OS-9対応。
5. 従来の当社製X68000シリーズ対応のHDU(ITX-403/203等)にも増設が可能です。(ただし従来の機種は1台目に固定)

※IT X640はMSX2 HD Interfaceにも接続することが可能。

■IT X640……………………………¥158,000
- 40MB HDU
- 平均アクセスタイム28ms

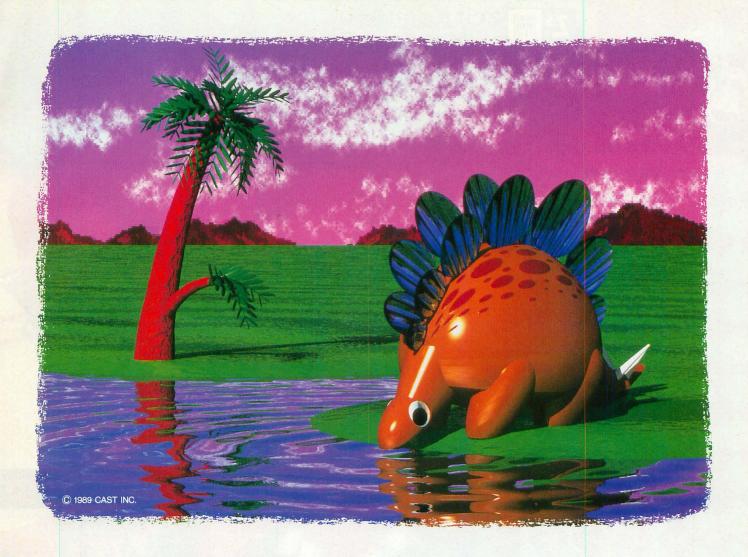
■IT X680……………………………¥198,000
- 80MB HDU
- 平均アクセスタイム20ms

※40MB×2の設定に固定されています。

アイテック株式会社

本社 〒550 大阪市西区新町1丁目3番12号 TEL:(06)532-0216 FAX:(06)532-7253
関西営業所 〒550 大阪市西区新町1丁目25番14号 TEL:(06)532-0120 FAX:(06)532-0543
関東営業所 〒275 千葉県習志野市谷津1丁目12番5号 TEL:(0474)77-7564 FAX:(0474)73-2759
名古屋営業所 〒460 名古屋市中区大須2丁目28番31号 TEL:(052)212-1487 FAX:(052)212-1627

インフォメーションセンター TEL:(06)532-0320

© 1989 CAST INC.

プロのための3次元コンピューターグラフィックス

C-TRACE

レイトレーシングソフトウェア

C-TRACE TOWNS	¥68,000
C-TRACE 68 (X68000対応)	¥68,000
C-TRACE 98DRY (PC-9801対応)	¥68,000
C-TRACE 98+ (PC-9801対応)	¥198,000
C-TRACE NEWS (SONY)	¥380,000
★C-TRACE 98TP	¥610,000
★C-TRACE 68TP	¥610,000

表示価格に消費税は含みません
★の製品は店頭販売いたしません 直接当社までお申し込みください。

フレーム・バァッファ

ディスプレイのマッハバンド(しま模様)が気になる方へ。
きれいなビデオ出力が欲しい方へ。1670万色同時表示、
C-FRAME68 新発売!!
フルカラー・フレームバァッファ、コンポジット入出力機能内蔵。
ペイントソフト付き ¥248,000
もちろん、C-TRACE68も対応。

▶これだけあれば後はいらない!?

CG98フルコース ¥138,000(PC-9801全機種対応)
フレーム・バッファ, ペイント, ポリゴン, レイトレ のフルセット
このセットでCGのすべての分野が体験できます。
※この製品は直接当社までお申し込みください。

株式会社キャスト 〒158 東京都世田谷区等々力2-1-13 TEL.03-705-0656 FAX.03-705-5224

回転体、円柱、面貫通処理などを利用し、

イメージを展開してみました。

BY DIGITAL CRAFT

2Dを携えれば、ー 自己表現の快感は、イ 画面の外にまで ン 揚々とひろがっている。グ。

X68000のハイ・ポテンシャル、65,536色モードに対応したサーフェイスモデルが、マウスだけで綿密に描ける3Dグラフィックソフト、ジーズ・トリフォニィ[デジタルクラフト]。スムース/フラット/ランダム、3種のシェーディングが目的に応じて使いこなせ、高度な面貫通や透明度機能、アンチエリアシング処理を実現。しかも、作成したモデルを最大7cut/secの視点移動により動かせるアニメーション機能を搭載。さらに32,768とすぐれたポリゴン数(3M増設時)を実現しました。プロ・バージョンの2Dグラフィックソフト、ジーズスタッフPRO-68Kとのツインユースで、新しいグラフィックスの悦楽にひたってください。

複雑な造形もオブジェクト毎に作り、

それらを組み合せることで

容易に、しかも正確に作成できます。

BY DIGITAL CRAFT

B I N A T I O N

Z's STAFF PRO-68K [Ver.2.0]

2Dのプロスペックソフト ジーズスタッフPRO-68K[Ver.2.0] ¥58,000税別

近未来MIDIオーディオ

Mu-1

Musicstudio ミューワン

1月末発売 ¥19,800(税別)

ミュージくん(MT-32)
ミュージ郎(CM-62、CM-32L)音源対応
データ曲集(98用5")コンバート機能搭載 MIDIプレイヤー

X68000/サイバースティック/CM-64/CF-10(デジタルフェダー)/MA-12C(マイクロモニター)

(特長)Mu-1は、ミュージくん(MT-32)、ミュージ郎(CM-64、CM-32L)音源対応のMIDIプレーヤーです。ソングファイルはもちろん、ミュージくん、ミュージ郎(ローランド98用5"2HD)のデータ曲集をそのままコンバートして聴いたり、リミックスすることができます。98曲連続演奏機能、ビジュアルインウィンドウ、サイバーステック対応など面白機能搭載。

ウィンドウ画面(ミュージックシェルII)

1. 音楽制作をスピーディにする、ミュージックシェルII搭載

2. "ミュージくん(ローランド)"、"ミュージ郎(ローランド)"、"Music PRO 68K & MIDI(シャープ)" "MML"のデータコンバート機能 ※Mu-1は、Musicstudio PRO68K(シャープ)と同じデータフォーマットです。

3. 98曲、自由な順番で連続演奏ができるチェインプレー機能搭載

ダブルクリックすれば簡単変換

4. 充実したリアルタイムミキシングエディット機能
   1)ボリューム、パンポットデータのミキシングエディット機能(ローランド社デジタルフェーダーCF-10にも対応)
   2)フェードイン、フェードアウトがコントロールできる2種類のグループフェダー搭載
   3)各トラックに再生用ベロシティーフェーダー搭載
   4)自由にエディットミキシングができる再生用キャンセルスイッチを各トラックに搭載
   5)オートフェーダー、オートパンポットエフェクト搭載

5. ステップ入力、エディット機能搭載 ※トラック単位のSave、Lode機能搭載

Copy、Paste等、チェイン編集は思いのまま

6. サイバースティックドラムパッド機能
   サイバースティックでドラム音源を鳴らすことができます。また音像の定位を自由にコントロールできるモードも搭載しています。

7. エクスクルーシブデータファイル機能付MIDIモニター搭載

8. 演奏しながらウィンドウ内でテレビや絵を呼び出せるビジュアル イン ウィンドウ機能搭載
   (*テレビ画面を楽しむには、シャープCZ-6VT1カラーイメージユニットが必要です)

"Magic Palette"グラフィックデータ表示

9. 落書きペイント搭載

10. 国本佳宏氏のオリジナル曲とKUNTA氏のイラストを収録

ハード構成――シャープ社製X68000本体、CRT、MIDIボード(CZ-6BM1)
ローランド社製MIDI音源、MT-32、CM-32L又は、CM-64

マジックパレットは販売(有)ミュージカル・フラン(開発サンミュージカルサービス)の超高速グラフィックエディタです。

オリジナルデータ曲集 ソングファイル 好評発売中 ¥5,800

SF-001 知恵ある暮しの味
国本 佳宏

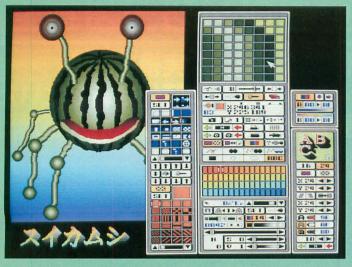
SF-002 インセクト
佐久間 正英

SF-003 ピーセズ・オブ・ワーク
本多 俊之

SF-004 あの娘のDNA
戸田 誠司

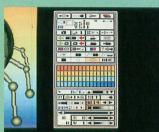
SF-005 リゾーム症候群
佐藤 允彦

SF-006 スケッチ
関根 安里

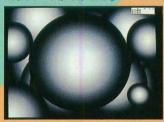
ソングファイルはアーティストによる世界初のオリジナルデータ曲集です。従来のカセットテープやCDでは音楽を聴くのみでしたがソングファイルは音色を変えたりデータを修正したりして好みのサウンドに創り変えることができます。あなたの感性で自由な音創りをして御聴きください。

X68000 ニューコンセプト&
超高速グラフィックエディタ

マジックパレット
MAGIC PALETTE

X68000シリーズ用 5"2HD GRX-1 ¥19,800
（価格に消費税は含まれておりません）

高品位なグラフィック機能を誇るX68000をターゲットに、新しいコンセプトにより設計開発された高性能かつ超高速なグラフィックツール"マジックパレット"。このツールは、あなたの豊かなイマジネーションをビジュアルにグラフィカルに演出する『魔法の絵の具』です。

超高速操作
●インスピレーションの流れを妨げない高速なウインドウ開閉。すべての描画&編集が超高速スピードで動作します。

●リアルタイム・グラデーションモニターにより、描画済みグラデーションを瞬時に別色のグラデーションにチェンジ。

スーパー・グラデーション&パレット
●強力なパレット機能による多彩なグラデーション表現。美しくて立体的な球面や虹などの多色グラデーションもワンタッチ。

●65,636色中255色のパレットをメモリ上に6種作成でき、切り替えられます。

驚異のコピー&ペイスト機能
●コピー元の絵はペイスト時にカーソルといっしょに動き、さらに、その下をすかして見ることもできるので正確な位置合せが得意です。

●透明機能のトリミングで必要部分のみを高速に連続ペイストができるので、コピー元の絵をまるでペン先のように扱って描くことができます。

描画テクニックとして使えるアンドゥー機能
●複数枚のメモリ画面をもっていて全画面のアンドゥーができ、アンドゥー画面の一部を切り出すなど新しいテクニックが使えます。

イメージユニット、イメージスキャナに対応
●イメージユニット、イメージスキャナを用いれば、テレビやビデオ、印刷物や写真などの画像を取り込んで編集したり、描画画面との合成ができます。
[対応機種：CZ-8NS1/GT-3000/GT-4000]

スプライトデータの作成と保存
●スプライトの作成と重ね合わせのシミュレーションが簡単にでき、スプライトデータとしてセーブできます。

データの互換性
●作成したデータは、X-BASICで読み込んで使えるほか、各種ツールで利用できます。

デモンストレーション機能
●描画、編集などの入力過程をそっくりディスクへ記録しあとで再現できるので、デモンストレーションを演出できます。

48ドットカラープリンタ対応
●48ドットカラープリンタをはじめ、以下のプリンタで印刷できます。
[対応プリンタ：CZ-8PK3/CZ-8PK4/CZ-8PC1/CZ-8PC4/CZ-8PN1/IO-735/PC-PR101、201シリーズ/VPシリーズ/HGシリーズ/APシリーズ]

その他の多彩な描画&編集機能
8、12、16、32、48ドットのCGROMフォントの文字入力（FPは増設メモリで使用可）/ベースタイル、タイルを外字として文字入力/文字間、行間、斜体、色指定の文字枠、長さと色設定による密着型文字影/40種のペン/エアーブラシ/ベースタイル（TLB）、タイル（TLM）、タイル（TLS）の作成と編集および重ね合わせ/スポイトによるカレントカラー設定/クロマキー機能およびフラッシング/2、4、8、16倍、高速移動&クリッピング可能なスコープ/拡大、縮小、上下反転、左右反転、ローテイト、90°回転、水平・垂直対象コピー、任意角度回転、スコープ移動ポインタとしても使える任意サイズの方眼とスケール表示/カラー16色、256色、216色、モノクロ16階調の各種イメージ取り込み/

画面&データ仕様
512×512ドット、256色モード/標準RAM時にメモリ画面を2枚、増設時に4枚、内1枚をコピーバッファで使用/パレット&HOOをクロマキー処理、&HO1をシステム内部で使用、&HO2-&HFFが色設定、色指定可能（65,536色より254色）/オリジナルなイメージデータフォーマット XBASICフォーマットデータ/16色、256色、240色のパレットデータファイル/スプライトキャラクタデータ/16×16×128の2色TLBベースタイル、16×16×128の15&16色タイル、16×16×128の255色タイルの各データファイル/

販売元：有限会社 **ミュージカル・プラン**
本社：〒107 東京都港区南青山3-14-14 サン南青山102 TEL.03(401)2751 FAX.03(401)1048
長野オフィス：〒380 長野県長野市居町1797番地 TEL.0262(24)3430

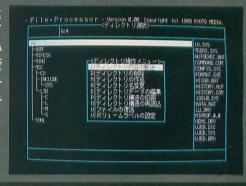
※お近くのパソコンショップ、楽器店にてお求め下さい。なお、入手が困難な場合は通信販売でお求めになることもできます。氏名・住所・電話番号・機種名を明記して現金書留にて当社迄お送りください。（消費税・送料サービス）

X68000シリーズ初！

ディスク操作では、絶大な実績を誇る 京都メディアがまた1つ時代を斬りさいた！ これは、必要だとか便利じゃない、快感だ!!

おめでとうございます。
貴方は、「ちょと待った！」という文句に誘われて他のページに進まずにこのページを読まれました。感謝の気持ちをこめて耳よりな情報をお教えします。98をも凌ぐといわれるX68000シリーズに新たに強力なユーティリティーが発売されます。このソフトは、68の事ならなんでも来いというPRO候補の方が使用されるとファイル管理ならなんでも来いの金棒になり、ビギナーの方が使用されると、いつのまにかHuman68kを使いこなせる様になってしまう魔法の力を持ったソフトです。

その機能を少しだけ紹介すると、ファイルのソートは、もちろんの事、ディレクトリの転送や、FATのエディット、削除してしまったファイルの復活にファイル属性の変更まで出来てしまいます。難しい事は、わかんないという貴方もだまされたと思って使ってみてよ！とにかくX68kユーザー必須アイテム。ぐちゃぐちゃになって管理が大変なフロッピーディスクやハードディスクにもういちど命を与えてみませんか？

写真は、開発中の98用画面です。

本格的ファイルマネージングソフトウェア　近日発売予定
X68000シリーズ用　28000円

The File Professor

THE FILE PROFESSOR動作条件

起動に必要な物：X68000本体，ディスプレイ，Human68k，(注)浮動小数点演算パッケージ
あると便利な物：X68k対応プリンタ，ハードディスク，増設RAM
(注) Human68kv.IIあるいは，C compilerに添付されています。

京都メディア

〒615 京都市右京区西院上今田町17-1 L&Pビル5F
TEL(075)822-3960(代)・FAX(075)822-3961

お待たせ致しましたバージョンアップ開始.!

発売開始

定価：¥22,000

エキサイティング・グラフィックツール　G68K VersionII-PRO

G68K Version II-PRO

OH! BUSINESS

●京都市山科区音羽西林町2
●京都市右京区西院上今田町17-1
サポート室：(075)502－2972
開発室：(075)822－4408

ご案内

この度、弊社では発売中のG68Kをバージョンアップ致します。つきましては、下記のとうりご案内させていただきます。

旧版G68Kは、お求めやすい価格と簡単操作により、入門用ツールとして多くのX68000ユーザーの皆様方よりご好評をいただいております。

今回のバージョンアップでは旧版の簡単操作を継承しつつ、業界でもトップレベルの処理スピードと前作を遥かに上回る、高機能・多機能・高速処理を実現致しました。

旧版G68Kユーザーの皆様方から頂いた多くのご意見を元に、本格的プロ仕様ツールとして大幅バージョンアップ致しました。

サンプルデータもプロのイラストレーターの手に依るコンピュータイラストを収録。また、専用グラフィックデータ集のシリーズ化も予定しております。

高速・高機能・低価格・1MB標準実装のメモリで完全に動作する本格派グラフィックツール。

■前作を大幅に上回る80種類のパレット
●自由に編集可能
●模様のついたパレットも作成可能
●HSV方式による色の合成
色相(色の種類)・彩度(色の濃さ)・明度(色の明るさ)
●簡単にお望みの色を作り出すための数々の機能を装備
●マスキング塗料・マスク除去塗料を装備
微妙な修正に威力を発揮
●2色の混合
●画面上より自由に色を取り込むスポイト機能
●パレット保存可能
●画面上より自由にタイルパターンを取り込むタイルパターン用カッターを装備
■32階調の濃淡をもつブラシ
●自由に形状を変更できるブラシが24種類
●ユーザーが自由に変更・ディスクに保存可能
■大幅に機能アップされたエアブラシ
●ブラシノズル口径、インク噴出速度・濃度を自由に設定
■32階調の濃淡を持つトーンパターン
●全てのペイントに有効。
●自由に変更・ディスクに保存可能
■強力な編集機能
●2倍、4倍、8倍に画面を拡大する拡大エディット機能（ルーペ機能）
●色を調整するカラーコレクタ
●任意角度の高速画像回転
●拡大・縮小
●左右・上下反転
●切り取りセーブ&ロード
●自由領域のコピー・移動
●標準実装のメモリで全画面が編集可能
■製図用具
●マスキング機能
●ペン描画時の直線
●指定領域のカラー変更
●円・楕円・ボックス・直線・自由領域
●これらの内部のペイント
●単色領域ペイント
■文字入力をサポート
●X68000標準24×24ドットキャラクタの表示
■外部機器のサポート
●豊富な対応周辺機器
■起動直前の画面を保存しながら起動することも可能
■UNDO機能（取り消し処理）
●ペイント等に失敗してもワンステップ前に戻ることが可能
■市販グラフィックツールとのファイルコンバーターが付属
●Z's STAFF-PRO 68Kとのファイル変換が可能
■ノンプロテクト
●ハードディスクへの転送も可能(自由インストール)
■FileはBASICのGL3形式
●BASICより簡単に読み出し可能

バージョンアップについて ▶登録ユーザーにはバージョンアップ案内を送付致します。

旧版G68Kをお持ちのユーザーの方でバージョンアップをご希望の方は、同封の申し込みハガキにてお申し込みください。

お申し込みは、バージョンアップ専用ハガキに限ります。(コピーは不可。)

バージョンアップに際して、旧版のG68Kの返却は必要ありません。今回のバージョンアップは、ディスク・マニュアルの交換バージョンアップではありません。

新版のパッケージ入製品(G68K VersionII-PRO)定価¥22,000をバージョンアップ価格¥10,000(送料込み)にてお届けいたします。旧版G68Kは入門用ツールとしてそのままお使い下さい。

同封のバージョンアップ申し込みハガキを必ずご使用になってください。シリアルNo.を必ずご記入下さい。

同封のDEMOディスクにてニューバージョンの機能を一部ご紹介させて頂きます。是非、ご覧ください。

また、ハガキにて今回お申し込みのユーザー様にはもれなく弊社オリジナルTシャツを商品発送時にお届け致します。

▶お問い合わせ・お申し込みは上記電話番号までお願い致します。(上記サポート室迄)

X68000版、勇気活性起爆装置、登場。

STAR COMMAND

スターコマンド

銀河系の彼方で、組織的宇宙海賊、コングロマリットの侵略が始まる

驚愕のグラフィックスで表現された、未知の世界が君を待っている

空間をうごめく不気味な宇宙船 彼等の真の目的とはいったい……？

敵基地の内部では、未だかって君が体験したことのないような出来事が

スターコマンド。それは、人類の平和を守るべく組織された宇宙防衛軍である。遥か銀河系の彼方で繰り広げられる、異星人との死闘ドラマ。今、神秘と未知の宇宙空間で、勇気あるスターコマンド達の、長く厳しい冒険の旅が始まる………。X68000版は、全てのグラフィックを刷新。X68000の機能を最大限に生かした、リアルでスリリングなグラフィックで、映画を観ているような大迫力の戦闘シーンが展開する。パワーアップしたFFG(フル・フレーム・グラフィック)方式、随所に取り入れられたアニメーション処理、あのすぎやまこういちによる、臨場感溢れるBGM。ドラマチックなシーンの数々が、君をファンタジック・スペースへと引き込んでいく。緻密に組み立てられたシナリオ、データ、そしてイベント。～人類の英知に挑戦する、覚醒RPGの誕生だ。 X68000版／ディスク4枚組

12月22日発売予定!!

予価 9,800円

上記の価格には、消費税は含まれておりません。

PC-9801版／好評発売中!!

HuLinks Inc.

株式会社ヒューリンクス

〒170 東京都豊島区東池袋3-7-7 福地ビル

TEL. 03(590)2311(代)

悪夢は終わっていない。

かつて世界は邪教神と悪魔とが人々を震撼させていた。
しかし、この地に住んでいた魔法族・レナムが邪教神を倒し
人々に平和を与えてくれた。
――それから1500年。
人々は戦いを忘れ、
レナム一族が消息を絶ったころ
邪教神バーディスが復活しようとしている。
レナムの力と、
聖剣ソル。そして…
邪教神を倒す術を求めて、
ソード・マスター：リーク・ラウエルは今、旅立つ。

SWORD OF LEGEND Lenam
レナム

広がる音、
深まる感動。
臨場感あふれる、
先進のMIDI音源
対応！
(X68000, PC-9800)

アドベンチャー・ロールプレイング・ゲーム《レナム》
●壮大な世界で繰り広げられるRPGの要素と、奥深いストーリーが楽しめるAVGの要素がドッキング！
●幅広いゲーム展開が楽しめる！
1キャラクター同士の会話等のストーリー展開とモンスターとの戦闘シーンはそれぞれ異なるコマンド・システムを採用。
2ストーリー展開でのキャラクターの移動はワールド・マップが表示。

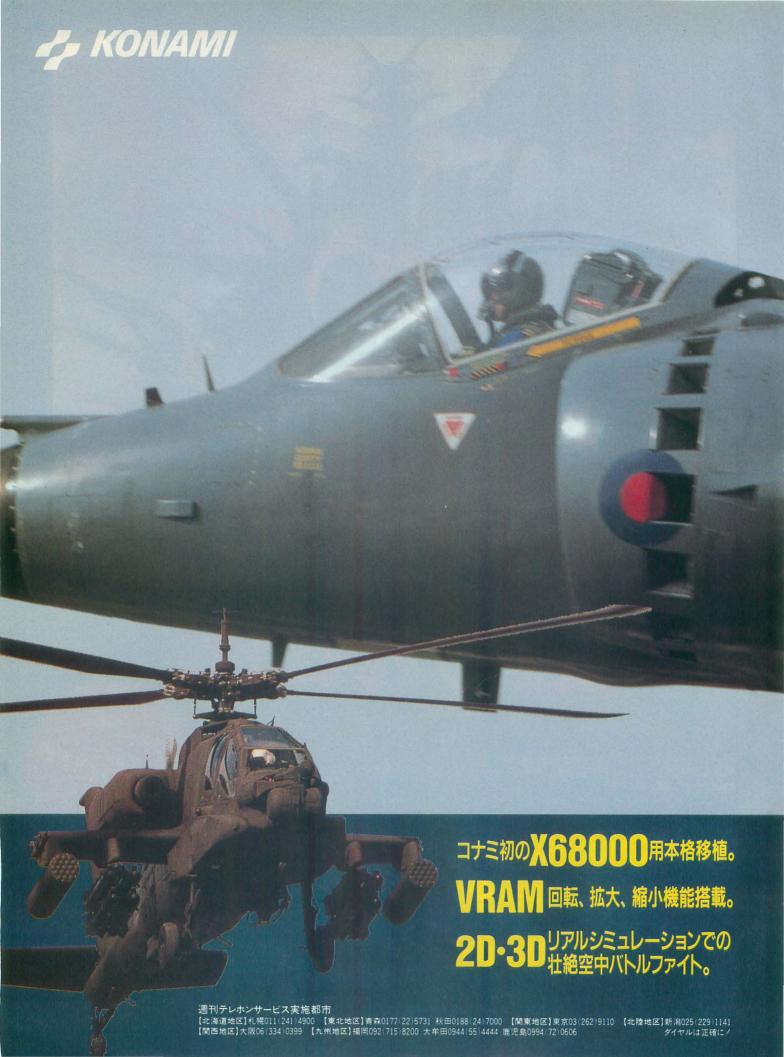
X68000／5"2HD（6枚組）／¥9,800（税別）
●Roland MT32 ＊MT32を使用する際はCZ-6BMIが必要です。／マウス対応（キーボード入力は不可）

PC-9800（Vシリーズ以降　要メモリ640KB）／3.5"　5"2HD（4枚組）／¥9,800（税別）
●FM音源対応／Roland MT32／マウス対応（キーボード、ジョイスティックも可）／
＊NEC純正MS-DOSが必要です。MS-DOSはマイクロソフト社の登録商標です。　**近日発売**

MSX 2/2+（V RAM128KB）／3.5" 2DD（4枚組）／¥8,800（税別）
●FM PAC対応／マウス対応

X68000
MSX 2/2+　**12月22日 同時発売!!**

株式会社ヘルツ　〒169 東京都新宿区北新宿区2-1-16松本ビル3号2F
TEL：03(371)3012／FAX：03(369)4071

KONAMI
コナミ初のX68000用本格移植。
VRAM回転、拡大、縮小機能搭載。
2D・3Dリアルシミュレーションでの壮絶空中バトルファイト。
週刊テレホンサービス実施都市
【北海道地区】札幌011(241)4900 【東北地区】青森0177(22)5731 秋田0188(24)7000 【関東地区】東京03(262)9110 【北陸地区】新潟025(229)1141
【関西地区】大阪06(334)0399 【九州地区】福岡092(715)8200 大牟田0944(55)4444 鹿児島0994(72)0606
ダイヤルは正確に／

人類は、最後の反撃を開始した。

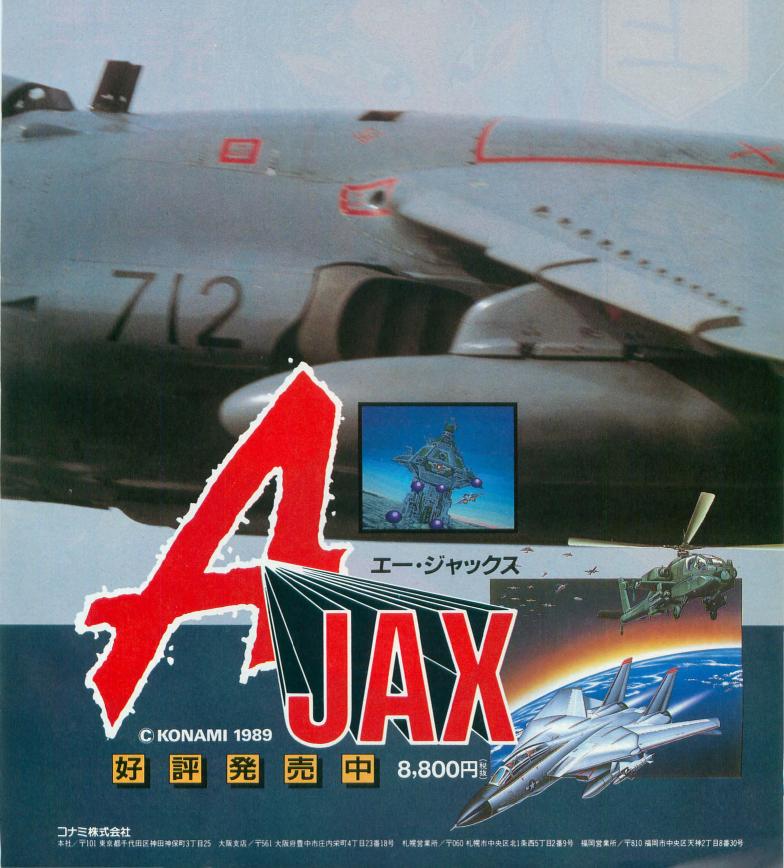
X68000の持つ能力の限界に挑戦し、アーケード版で体験した、あの興奮と感激をあますところなく再現した本格バトル・シミュレーション。

Human
Hudson

上海 II

麻雀的頭脳遊戯。

タイトルでは龍のイラストが今にも画面から飛び出して来そうな気配だ。

かつてないリアリティで迫る
X68000版
12月新登場!
希望小売価格 ¥6,800

SHANGHAI II
A captivating strategy challenge derived from the ancient Chinese game of Mah Jongg
ACTIVISION
ENTERTAINMENT SOFTWARE

コンピューターとの頭脳の勝負！給柄を合わせすべての牌を取り除け！

麻雀牌を使ったパズルゲームの名作！！ あの「上海」がバージョンアップして×68000に新登場！！ 5段に積まれた144個の麻雀牌を同じ給柄に合わせて端から取っていき、すべて取り終わればクリアできる。牌の積み方も6パターン、難易度も4段階に切り替えられるぞ。さらにスコア表示（牌1個につき1点）を導入し、ベスト10は記憶され点数を競い合うこともできる！
キミの頭脳をフル回転させ、難解難題を打ち破れ！！

本社 〒062 札幌市豊平区平岸3条5丁目1番18号 ハドソンビル TEL 011-841-4622
東京支社 〒162 東京都新宿区市谷田町3丁目1番1号 ハドソンビル TEL 03-260-4622
大阪支店 〒542 大阪市中央区東心斎橋1丁目1番10号 大阪料理会館ビル5階 TEL 06-251-4622
営業所 札幌・名古屋・福岡

METAL SIGHT

MIDI対応

メタル
サイト　超弩級3Dシューティングゲーム

扉が開かれる……

X68000は、限界を知らない…
画面に入り切らないボスキャラ…
動態視力の限界に迫るスピード…
超高速疑似スプライトが織り成す
メタルサイトワールド！

要塞の中を…サバンナを…
洞窟の中を…天空を…

そして…
宇宙を…

XF-068A　クロスドッグが駆け抜ける
華麗なミュージックにのって今…
TAKE OFF！

A.D.1989.12. 夢が形になる…

スタッフ募集中!!

開発元
Team Cross Wonder
〒203 東京都東久留米市本町1-13-11
TEL.0424-74-6009

ONLY X68000　標準価格 **8,800**円

SYSTEM SACOM & TEAM CROSS-WONDER

NOVEL WARE

A DAY IN THE LIFE OF 2049
FILE 1

38万キロの虚空

ノベルウェアの更なる進化

西暦2049年。人口の増加で地球は飽和状態をむかえていた。この限界状況を打破するために、アメリカが主導し世界が共同で開始した事業、それが宇宙植民計画である。これは地球人類を宇宙空間に移住させるというものであり、一歩一歩計画は進められてきた。そして2049年、計画の最初の成果SCが遂に完成した。ストーリーはこのSCのオープンをめぐるサスペンスである。プレイヤーはSC取材班の一員"相馬謙"となる。SCに対する様々な疑問を持ちながら相馬謙は地球からSCへの旅、SCの取材ツアーなど、次々と未知の世界を目の当たりにしてゆく。相馬謙はこの計画についてほとんど知識がなく、プレイヤーと同じ程度の知識で取材に挑む。彼が見て感じるものこそがプレイヤーの感じるものなのだ。そしてこの取材活動の中で、SCオープンを巡る陰謀の影が……SCに反対するも賛成するもプレイヤー次第。2049年の世界をじっくり楽しんでもらいたい。

●好評発売中!!
●X68000　5-2HD
●ローランド社MT-32完全対応
（(MIDIインターフェイスボードCZ-6BM1が必要です。)
●標準価格 9,800円

FM-TOWNS 新発売!!
標準価格 **9,800**円

MIDI対応

SACOM
株式会社 システム サコム
〒130 東京都墨田区両国4-38-16
両国桜井ビル4F
TEL.03-635-7609

＊標準価格には消費税は含まれておりません。

1989→1990 TSUKUMO'S YEAR

ツクモより感謝をこめて ツクモ

X68000のお店ならここ！ツクモ7号店2F！そしてツクモ通信販売部☎(03)251-9911

通信販売のお申し込みは料金無料の

フリーダイヤル！受注専用

☎0120-377-999

商品についてのお問い合せは各店又は
通信販売部☎(03)251-9911

今年最後の運だめし

「コンピュータルーレット」で
豪華な景品を当てよう！
好評続行中12/31迄 東京地区のみ
1万円以上お買物の方に1回のチャンス！10万円以上の場合は10万円毎に1回追加！

秋葉原電気まつり 賞金総額7000万円

THE AKIHABARA EXCITING FESTA

毎年恒例秋葉原電気まつり。
'89.11/24(金)〜'90.1/7(日)まで！
★5,000円以上のお買物でもれなく抽選券をプレゼント！

ゲームソフトが安い！

ツクモ各店でゲームソフト25%offセールを開催！
期日：12/23・24・25日：'90.1/2〜7

X1 turbo Z III セット ツクモ特価 ¥158,000

- CZ-888C-BK………¥169,800
- CZ-860D-BK………¥92,200

モデム

オムロン MD-2400B 定価¥49,800
(300/1200/2400ボー)
特価¥17,800 台数限定

今、大容量のハードディスクが大人気！

※1D番号の切り替えスイッチにより増設ドライブとしても使用可能。

アイテック
IT X-640 40MB、28ms
定価¥158,000
ツクモ特価¥128,000

NEW

IT X-680 80MB(40MB+40MB)、20ms
定価¥198,000
ツクモ特価¥158,000

大特価販売中！

X68000オリジナルグッズコーナーも増えて更に人気上昇中！

特価販売中

X68000 EXPERT / EXPERT HD
PERSONAL WORKSTATION
CZ-602C(縦置タイプ2M RAM標準搭載)
……………………定価¥356,000
CZ-612C(40MBハードディスク内蔵タイプ)
……………………定価¥466,000

X68000シリーズ 好評発売中!!

特価販売中

X68000 PRO / PRO HD
PERSONAL WORKSTATION
CZ-652C(横置タイプ1M RAM標準搭載)
……………………定価¥298,000
CZ-662C(40MBハードディスク内蔵タイプ)
……………………定価¥408,000

アクセサリーいろいろ(税別)

- ★ツクモオリジナルキーボード延長ケーブル…ツクモ特価¥1,980
- ★キーボードシリコンカバー……ツクモ特価¥2,400
- ★キーボードセフティカバーASC108…ツクモ特価¥2,400
- ★キーボードダストカバーADC108……ツクモ特価¥1,000

NEW MIDIセット

Aセット

CM-32L MIDI音源(MT-32コンパチ)…………定価¥69,000
SX-68M MIDIボード……………………定価¥19,800
CZ-247MS MUSIC PRO-68(MIDI)……定価¥28,800
合計定価¥117,600
ツクモ特価¥99,800(消費税別途¥2,994)

Bセット

CM-32L MIDI音源(MT-32コンパチ)…………定価¥69,000
SX-68M MIDIボード……………………定価¥19,800
CZ-252MS MUSICstudio PRO-68K Ver1.1…定価¥28,800
合計定価¥117,600
ツクモ特価¥99,800(消費税別途¥2,994)

Cセット

CM-64 MIDI音源(MT-32+サンプリング音源)…定価¥129,000
SX-68M MIDIボード……………………定価¥19,800
CZ-252MS Musicstudio PRO-68K Ver1.1…定価¥28,800
合計定価¥177,600
ツクモ特価¥153,000(消費税別途¥4,590)

周辺機器

CZ-6BE1 1MB内蔵RAM(CZ-600C専用)……………定価¥35,000
CZ-6BE1A 1MB内蔵RAM(ACE・PROシリーズ専用)……定価¥38,000
CZ-6BE2 2MB増設RAMボード……………………定価¥79,800
CZ-6BE4 4MB増設RAMボード……………………定価¥138,000
CZ-6BP1 数値演算プロセッサボード……………………定価¥79,800
※大好評インテリジェントコントローラー登場！
これであなたの部屋はゲームセンター…。
CZ-8NJ2 ……………………標準定価¥23,800

やっぱり、カラープリンターが欲しくなる

- カラー漢字24ドット熱転写プリンター
  CZ-8PC3……………ツクモ特価¥49,800
- カラー漢字48ドット熱転写プリンター(色は黒、又はグレーを指定して下さい。)
  CZ-8PC4……………定価¥99,800
- カラーイメージジェットプリンター
  IO-735X……………定価¥248,000
- 24ピンカラー漢字ドットインパクトプリンター(80桁)
  CZ-8PG1……………定価¥130,000
- 24ピンカラー漢字ドットインパクトプリンター(136桁)
  CZ-8PG2……………定価¥160,000

お勧めソフトウェア

Kamikaze(神風) 統合型スプレッドシート…ツクモ特価¥57,800
SOUND PRO-68K サウンドエディタ……………定価¥15,800
MUSIC PRO-68K ミュージックツール……………定価¥18,800
Sampling PRO-68K AD PCM活用ソフト……………定価¥17,800
Musicstudio PRO-68K V1.1 MIDIマルチレコーディングソフト
……………定価¥28,800
MUSIC PRO-68K(MIDI) MUSIC PRO-68KのMIDI版
定価¥28,800
ソングライブラリ〈101曲集〉 MUSIC PRO-68Kデータ曲集 定価¥ 8,800
Communication PRO-68K 通信ソフト……………定価¥19,800
た〜みのる 通信ソフト……………ツクモ特価¥10,900
た〜みのる2 「た〜みのる」Ver UP版……………ツクモ特価¥15,200
DATA PRO-68K リレーショナルデータベース……………定価¥58,000
CARD PRO-68K カード型データベース……………定価¥29,800
※CYBERNOTE PRO-68K 複合型パーソナルデータベース
……………定価¥19,800
※Stationery PRO-68K ステーショナリーツール…定価¥14,800
※電子手帳との通信ケーブルは別売です。(¥2,500)
システム手帳リフィル集 CARD PRO-68K用フォーム集…定価¥9,800
活用フォーム集 CARD PRO-68K用フォーム集……………定価¥9,800
Z's STAFF PRO-68K Ver2.0 グラフィックツール
……………ツクモ特価¥49,300
Z's Triphony DIGITAL CRAFT
3次元サーフェイスモデリングツール……………ツクモ特価¥33,800
New Print Shop PRO-68K 高機能ポップアートツール
……………定価¥19,800
サイクロン Express 2.0 レイトレーシングソフトウェア
……………ツクモ特価¥67,000
Terazzo SPRITE EDITOR PRO-68K
高性能スプライトエディタ……………ツクモ特価¥16,800
C-TRACE 68 レイトレーシングソフトウェア……ツクモ特価¥57,800
C COMPILER PRO-68K C言語開発セット……定価¥39,800
Final X68000 マルチファイル・スクリーン・エディタ ツクモ特価¥32,300
AI-68K AIプログラム開発ツール……………定価¥188,000
REDUCE 数式処理用ソフト……………ツクモ特価¥195,000
OS-9/X68000 X68000用OS-9……………定価¥29,800
C & プロフェッショナルパッケージ OS-9/X68000用Cコンパイラセット
……………ツクモ特価¥49,300
mFORTH Compiler FORTHコンパイラセット
ツクモ特価¥17,800
Human68K Ver2.0 Human68KのNEWバージョン……定価¥9,800
※その他、ゲームソフトも続々発売中ですので、詳しくはお尋ね下さい。

★超特価はお電話にてお問い合せ下さい。

㊠AM10:15〜PM7:00 年内無休㊡1/10・17

ツクモは「スーパーX PRO SHOP」です。

PRO STAFF ツクモ

九十九電機㈱ 〒101-91 東京都千代田区神田郵便局私書箱135号

ツクモ7号店 ☎03-253-4199

便利で安心な通信販売
通信販売部☎03-251-9911

- ツクモ5号店 ☎03-251-0531
- ニューセンター店 ☎03-251-0987
- 名古屋1号店 ☎052-263-1655
- 名古屋2号店 ☎052-251-3399
- ツクモ札幌 ☎011-241-2299

カード払い	全国代金引き換え配達	クレジット払い	現金書留払い	銀行振込払い
通信販売での御利用カード：ツクモグローバルカード、VIPカード、セントラル、ジャックス 御本人様より電話で通信販売部へお申し込み下さい。	お申し込みは☎03-251-9911へお電話1本！ 配達日の指定もできます。	月々¥3,000以上の均等払いも頭金なし。 夏・冬ボーナス2回払いも受付中。	〒101-91 東京都千代田区神田郵便局私書箱135号 九十九電機㈱通信販売部Oh!X係	事前に☎でお届け先をご連絡下さい 富士銀行 神田支店(普)No.894047

★表示価格には消費税は含まれておりません。

超低金利クレジットOK!!

またまた秋葉原でおなじみの

●お近くの方はお
●本体単品で特
●ビジネスソフト定

12/15〜1/15

CYBER STICK
● CZ-8NJ2
(定価¥23,800)
超特価!!
▶価格はTEL下さい

X-1ターボZⅢ 特別ご提供品!! 台数限定

● CZ-888C＋CZ-860D＋M-2HD(10枚)
定価¥269,600▶特価¥164,800
(ボーナス併用も有りますTEL下さい)

(・ジョイカード
・ゲーム3種
・パソコンラック(A)3段
プレゼント中)
送料消費税込み!!

12回	14,300	24回	7,500	36回	5,100	48回	4,000	60回	3,300

ジョイスティック 送料¥500
● X-1PRO
定価¥9,500▶特価¥7,800
● ASCII STICK
定価¥6,800▶特価¥5,500

P&A超低金利クレジットをご利用ください!!

X68000EXPERT & EXPERT-HD (送料消費税込み)

EXPERT&PROセットでお買い上げの方に
● ディスケット(10枚)
● ゲーム
● アフターバーナー(定価¥9,200)
● CZ-8NJI(ジョイカード)
プレゼント中

EXPERT

(ボーナス併用も有ります。TEL下さい)

Ⓐセット：CZ-602C＋CZ-603D……定価¥440,800▶現金価格はお電話下さい

12回	28,100	24回	14,700	36回	10,100	48回	7,900	60回	6,600

Ⓑセット：CZ-602C＋CZ-602D……定価¥455,800▶特価(現金価格はお電話下さい)

12回	29,200	24回	15,300	36回	10,500	48回	8,100	60回	6,800

Ⓒセット：CZ-602C＋CZ-612D……定価¥475,800▶特価(現金価格はお電話下さい)

12回	30,600	24回	16,000	36回	11,000	48回	8,600	60回	7,200

Ⓓセット：CZ-602C＋CU-21CD……定価¥495,800▶特価(現金価格はお電話下さい)

12回	31,500	24回	16,500	36回	11,400	48回	8,800	60回	7,400

EXPERT-HD

Ⓐセット：CZ-612C＋CZ-603D……定価¥550,800▶特価(現金価格はお電話下さい)

12回	35,000	24回	18,300	36回	12,600	48回	9,800	60回	8,200

Ⓑセット：CZ-612C＋CZ-602D……定価¥565,800▶特価(現金価格はお電話下さい)

12回	36,100	24回	18,900	36回	13,000	48回	10,100	60回	8,400

Ⓒセット：CZ-612C＋CZ-612D……定価¥585,800▶特価(現金価格はお電話下さい)

12回	37,500	24回	19,600	36回	13,500	48回	10,500	60回	8,700

Ⓓセット：CZ-612C＋CU-21CD……定価¥605,800▶特価(現金価格はお電話下さい)

12回	38,400	24回	20,100	36回	13,800	48回	10,800	60回	9,000

X68000PRO & PRO-HD (送料消費税込み)

EXPERT&PROセットでお買い上げの方に
● ディスケット(10枚)
● ゲーム
● アフターバーナー(定価¥9,200)
● CZ-8NJI(ジョイカード)
プレゼント中

PRO

(ボーナス併用も有ります。TEL下さい。)

Ⓐセット：CZ-652C＋CZ-603D……定価¥382,800▶特価(現金価格はお電話下さい)

12回	24,500	24回	12,900	36回	8,900	48回	6,900	60回	5,800

Ⓑセット：CZ-652C＋CZ-602D……定価¥397,800▶特価(現金価格はお電話下さい)

12回	25,600	24回	13,400	36回	9,200	48回	7,100	60回	6,000

Ⓒセット：CZ-652C＋CZ-612D……定価¥417,800▶特価(現金価格はお電話下さい)

12回	27,000	24回	14,200	36回	9,700	48回	7,600	60回	6,300

Ⓓセット：CZ-652C＋CU-21CD……定価¥437,800▶特価(現金価格はお電話下さい)

12回	28,000	24回	14,700	36回	10,000	48回	7,800	60回	6,500

PRO-HD

Ⓐセット：CZ-662C＋CZ-603D……定価¥492,800▶特価(現金価格はお電話下さい)

12回	31,400	24回	16,400	36回	11,300	48回	8,800	60回	7,400

Ⓑセット：CZ-662C＋CZ-602D……定価¥507,800▶特価(現金価格はお電話下さい)

12回	32,400	24回	17,000	36回	11,700	48回	9,100	60回	7,500

Ⓒセット：CZ-662C＋CZ-612D……定価¥527,800▶特価(現金価格はお電話下さい)

12回	33,900	24回	17,800	36回	12,200	48回	9,500	60回	7,900

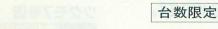
Ⓓセット：CZ-662C＋CU-21CD……定価¥547,800▶特価(現金価格はお電話下さい)

12回	34,800	24回	18,200	36回	12,500	48回	9,700	60回	8,100

X68000PRO/ACE-HD〜P&Aスペシャルセット＝限定誌上販売!!

35%OFF　送料、消費税別

X-68000PRO 特別ご提供品

台数限定

● CZ-652C(本体)
● CZ-611D(モニター)
● CZ-8PK8(24ピン、漢字、136桁)

＋ (● ジョイカード(8NJ1)
● ディスケット(10枚)
● ゲームプレゼント中)

(定価¥584,000) 特価¥378,000

12回	32,900	24回	17,200	36回	11,800	48回	9,200	60回	7,600

(ボーナス併用も有ります。TEL下さい。)

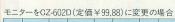
X-68000ACE-HDセット(台数限定)

● CZ-611C(本体)
● CZ-603D(モニター)
● CZ-8NJ2(CYBER STIC)

＋ (● ディスケット10枚
● ゲーム
● 送料、消費税込み)

定価¥508,400 P&A超特価!価格はお電話下さい。

12回	28,700	24回	15,000	36回	10,300	48回	8,000	60回	6,700

モニターをCZ-602D(定価¥99,88)に変更の場合

12回	30,100	24回	15,700	36回	10,800	48回	8,400	60回	7,000

● CZ-612D(定価¥119,800)に変更の場合

12回	31,300	24回	16,400	36回	11,300	48回	8,700	60回	7,300

● CZ-611D(定価¥145,000)に変更の場合

12回	30,700	24回	16,100	36回	11,000	48回	8,600	60回	7,100

(ボーナス併用も有ります。TEL下さい。)

● 本広告の掲載の商品の価格については、消費税は含まれておりません。
● 営業時間＝平日AM10:00〜PM8:00、日祭AM10:00〜PM7:00

年末年始のご案内：12/30〜1/3はお休みさせていただきます。

JAPANESE EDITION

全米No.1パソコン誌

12月18日創刊

定価580円

毎月18日発売

FIRST LOOKS

- 最新情報Lotus 1-2-3 Rel. 3.0
- NECカラー液晶ラップトップPro Speed CSX
- エプソンPC-286UX／LFテスト速報
- 着脱式HDD POKEDYに40Mタイプ登場
- 7,000円の超低価格ワープロEG Light試用レポート

『パソコン・マガジン』は
PCラボのベンチマークテストをもとに
総合的な製品評価記事をわかりやすくお届けします。

多種多様のパソコン製品が氾濫する現代において、ユーザーは、どの製品を、どのような基準で選ぶべきかという大きな問題を抱えています。そこで、製品購入の指針を提供するバイヤーズガイド(テストマガジン)として「パソコン・マガジン」はお応えします。総合評価テスト記事を中心に、新製品情報、新製品トレンドを語り、さらに技術、製品を評価する視点を養うようなコラムなどで構成します。

創刊記念特別付録
オリジナル・ディスクラベルシート

日本ソフトバンク出版事業部
東京都千代田区九段南2-3-26 TEL03(230)7670：営業部

1989年度 GAME OF THE YEAR ノミネート作品発表

1989年度のゲームソフトの傾向と対策

早いもので，このGAME OF THE YEARがスタートしてもう5年。最初のころは，ゲームの数も少なくて困ったものでしたが，いまや逆にゲームがあふれかえっていて，ノミネート作品を選ぶのが大変なくらいです。まあ，ゲーマーにとってはうれしい限りですけどね。

さて，1980年代最後の今年は，やはりX68000が主流となってしまいました。昨年からそのきらいはあったものの，いざこうしてフタを開けて現実を目の当たりにしてしまうと感慨深いものがあります。パソコンゲームの流れというのはそんなものだと頭ではわかっていても，やはり寂しいものですね。

そんなわけで，今年はOh!MZ賞，Oh!X1賞，そしてOh!X68賞という各機種別の賞は取り止めることにしました。その代わりといってはなんですが，新たにシミュレーションゲーム賞というものを設置しました。次次と出てくるゲームの数々，当然その分だけそれぞれのゲームのジャンルも充実してきました。こういった時代の流れに対応すべく，新しい部門賞を設置していくことは，パソコンゲーム全体の活性化につながるのでは，と思い設置したわけです。

それにしても，この1年のパソコンゲーム界は充実していたのではないかと思います。X68000におけるビデオゲームからの移植の応酬，それに対するX1の名作たち。その両方の力がぶつかり合って，ゲームを作る側としてはより高度なレベルのものを作ることを強いられ，昨年よりも質，量ともに向上したといえるでしょう。こういった状況は喜ばしいことだし，今後も大歓迎です。

右の表を見てください。これは，昨年と同じようにアンケートハガキの「推薦するソフト」の欄をまとめたものです。2月号から12月号までの11カ月間のアンケートハガキのなかから，各月1100通ずつ，計12100通を無作為に選びその結果を集計してあります。

見ていただければわかるとは思いますが，X68000の「アフターバーナー」がダントツの1位。そのほかX68000では，アクション，シューティングが上位を占めています。一方X1は，昨年もノミネートされていたゲームが名前を連ねています。さあ，一発逆転なるか，皆さんの各賞への清き一票をお待ちしております。

1989年 GAME OF THE YEAR アンケートベスト50

1. アフターバーナー（X68000）……937
2. テトリス（X1/X68000）……486
3. ソーサリアン（X1）……479
4. ジェノサイド（X68000）……453
5. SUPER大戦略（X1/X68000）……303
6. スタークルーザー（X1/X68000）……272
7. ファンタジーゾーン（X68000）……266
8. 三国志（X1/X68000）……191
9. ドラゴンスピリット（X68000）……184
10. ウィザードリィ（X1/MZ-2500）……181
11. イースII（X1）……180
12. 信長の野望・戦国群雄伝（X1）……176
13. サンダーフォースII（X68000）……173
14. ラストハルマゲドン（X68000）……149
15. リボルティーII（X1）……132
16. 今夜も朝までPOWERFULまあじゃん2（X1/X68000）110
17. Might & Magic2（X1/X68000）……103
18. R-TYPE（X68000）……99
19. アドヴァンスト・ファンタジアン（X1）……90
20. ボスコニアン（X68000）……82
21. めぞん一刻・完結編スペシャル（X68000）……81
22. スペースハリアー（X68000）……78
23. デス・ブリンガー（X68000）……77
24. マーダークラブDX（X68000）……68
25. ニュージーランドストーリー（X68000）……66
26. サイオブレード（X1）……61
27. パックマニア（X68000）……59
28. 源平討魔伝（X68000）……58
29. Might & Magic（X1/X68000）……57
30. サバッシュ（X68000）……56
31. D-RETURN（X68000）……48
32. 水滸伝（X1）……44
33. ローグアライアンス（X1/X68000）……43
34. 琥珀色の遺言（X1/X68000）……41
35. イース（X1）……39
36. A列車で行こうII（X68000）……38
37. 信長の野望・全国版（X68000）……37
38. ハイドライド3（X1）……36
39. グラディウス（X68000）……35
40. 熱血高校ドッジボール部（X68000）……34
41. リングマスター1（X68000）……33
42. 大海令（X68000）……31
43. 沙羅曼蛇（X68000）……30
44. 上海（X68000）……29
45. ウイングス（X68000）……28
46. メタルサイト（X68000）……27
    野球道（X1）……27
48. 蒼き狼と白き牝鹿・ジンギスカン（X1/MZ-2500）……26
49. 太平洋の嵐DX（X68000）……25
50. ナイトアームズ（X68000）……24

Oh!Xゲーム大賞

この賞は，文字どおりいままでに発売されたゲームのなかで，最もプレイヤーを楽しませてくれたものに贈られる栄誉ある賞です。今年も多種多様なジャンルから15本がノミネートされました。これらのノミネート作品を見て気づくことは，X68000は今年発売されたシューティングが主流であるのに対し，X1は昨年もノミネートされたRPGが依然として人気が衰えず登場しているという点です。特に「イースII」と「ソーサリアン」の強さにはもう脱帽モノですね。

しかし，ちょっと残念に思う点は，相変わらず全体的に他機種からの移植ものがやたらと多いことです。そんななかで新規参入会社であるズームのオリジナルゲーム「ジェノサイド」のノミネートはうれしい限りです。日本ファルコムがゲーム界の五木ひろしなら，ズームは光GENJIといったところでしょうか。どこまで票を集めてくれるか楽しみです。

- アフターバーナー（X68000）
- イースII（X1）
- ウィザードリィ（X1/MZ-2500）
- 三国志（X1/X68000）
- サンダーフォースII（X68000）
- ジェノサイド（X68000）
- スタークルーザー（X1/X68000）
- SUPER大戦略（X1/X68000）
- ソーサリアン（X1）
- テトリス（X1/X68000）
- ドラゴンスピリット（X68000）
- 信長の野望・戦国群雄伝（X1）
- ファンタジーゾーン（X68000）
- ラスト・ハルマゲドン（X68000）
- リボルティーII（X1）

ゲームデザイン賞

この賞は，ゲームを作るうえでの発想の良さや，その基本設計を評価するものです。つまり，グラフィックやシナリオがどうのこうのというのではなく，ゲーム自体のコンセプトがよいもの，もしくは機種の特性を生かしたものに贈られる賞なのです。ここでは各機種の性能の違いなのか，X68000はアクションやシューティングといった動きのあるものが多く，X1はRPGやシミュレーションといったじっくり腰を据えてやるゲームがノミネートされました。

- アフターバーナー（X68000）
- 三国志（X1/X68000）
- ジェノサイド（X68000）
- ソーサリアン（X1）
- ファンタジーゾーン（X68000）

グラフィック賞

ゲームを買うときにはグラフィックで決める，という人がいるほど重要な要素であるグラフィック。単に絵がきれいなだけでなく，「きれいに見せる」といったセンスやテクニックもポイントとなります。

今年は，「速いスクロールをしても絵がちらつかない」などの利点を持ったX68000がやはり強く，4本もの作品がノミネートされました。しかし，じっくりと絵を楽しめる「イースII」も捨てがたいものがあります。

- ファンタジーゾーン（X68000）
- ジェノサイド（X68000）
- サンダーフォースII（X68000）
- マーダークラブDX（X68000）
- イースII（X1）

テーマ音楽賞

ゲームの面白さやそのゲームらしさを演出してくれるゲームミュージック。年々レベルが上がっていて，MIDI対応のものまで登場するなど，ゲームに浸るためには必要不可欠な要素となっているようです。

今年のノミネート作品は，血が躍るシューティングゲームにおいてさらに熱くさせてくれる，そんなBGMタイプのゲームミュージックが目立ちました。

- ジェノサイド（X68000）
- 新九玉伝（X1）
- スタークルーザー（X1/X68000）
- スターシップランデブー（X68000）
- ボスコニアン（X68000）

特殊演出部門賞

この賞は，ゲームの良し悪しにかかわらず，いかにしてそのゲームをプレイヤーにアピールしたか，ということを評価するものです。シナリオを損なわずに「ゲームを見せる」という工夫や技術，そしてプレイヤーをあっといわせるような演出を凝らしたゲームに贈られます。

今回ノミネートされた4作品は，それぞれ趣向を凝らした演出で，プレイヤーを楽しませてくれました。

- サイオブレード（X1）
- デス・ブリンガー（X68000）
- スタークルーザー（X1/X68000）
- ねじ式（X68000）

オリジナルシナリオ賞

ゲームは架空の世界だとはいえ，やはり背景にあるシナリオがしっかりしていないと，楽しみも半減してしまいます。特にRPGやAVG，シミュレーションといったゲームでは，シナリオのレベルの高さが，そのままゲームの良さにつながってきます。しかしながら，完成度の高さは年を追うごとに上がっているので，プレイヤーとしてはうれしい限りです。

さて，今回のノミネート作品のなかでもひときわ異彩を放っているのがツァイトの「ねじ式」。地味ながらもなにか奥深いものを感じさせる新しいタイプのシナリオでした。

- アドヴァンスト・ファンタジアン（X1）
- 水滸伝（X1）
- ねじ式（X68000）
- 原宿アフターダーク（X1）
- マーダークラブDX（X68000）

主演&助演キャラクタ賞

今年のこの部門では，人それぞれの思い入れというものを尊重して，あえてノミネートはしませんでした。今年1年あなたが出会ったパソコンゲームのなかで，忘れられない，忘れたくない，といったキャラクタたちにぜひ1票を投じてあげてください。

ノミネートなし

SF&ファンタジー賞

昨年はお休みさせていただいたこの賞ですが，またまた復活させることになってしまいました。理由はノミネート作品を見ていただければわかると思います。

架空の存在であるゲームの世界にあって，どれだけ独自のSFらしく，またファンタジックな世界を展開してプレイヤーの想像力を沸き立たせるか。これは古今東西ゲームにおいての重要な課題です。「ああ，このゲームをやってよかった」，そう思わせるようなドラマチックな展開があるSF&ファンタジーゲームにこそ贈りたい賞です。今回ノミネートされた4作品は，そういった点で十分にこの賞に値すると思います。

「ねじ式」が？　とおっしゃる方もいるとは思いますが，あれはあれですごくファンタジックなゲームだと思い，ノミネートさせてもらいました。

- 38万キロの虚空（X68000）
- Might & Magic II（X1/X68000）
- アドヴァンスト・ファンタジアン（X1）
- ねじ式（X68000）

インテリジェント賞

昨年パズルゲーム賞とテーブルゲーム賞に分けたこの部門でしたが，テーブルゲームのノミネートの大部分が麻雀ゲームという事態がまたもや生じてしまいそうだったので，麻雀ゲームのレベル向上を願ってまたくっつけてしまいました。

頭の血のめぐりをよくしたり，のほほんと楽しむのにはこれらの作品がぴったりだと思いこの5作品をノミネートしました。

- 琉球（X68000）
- テトリス（X1/X68000）
- 雀豪1（X68000）
- 森田将棋II（X68000）
- 麻雀狂時代SPECIAL II・冒険編（X1）

移植ビデオゲーム賞

この賞は，ビデオゲームから移植されたゲームのなかでも完成度が高く，人気も高かったものに贈られます。さすがにビデオゲームの主流がシューテイングとアクションのせいか，X68000のひとり舞台と化してしまったようです。ゲーセンで100円玉を注ぎ込んだゲームたちが，すべて居ながらにしてできる幸せな時代が，もうそこまで来ているようですね。

- R-TYPE（X68000）
- アフターバーナー（X68000）
- ファンタジーゾーン（X68000）
- パックマニア（X68000）
- ニュージーランドストーリー（X68000）

シューティングゲーム賞

今年のこの部門は豊作だったので，ノミネート作品を選ぶのにずいぶんと頭を悩ませました。が，結局無難な線で落ち着いたといえましょう。やはり，シューティングという部門のせいか，X68000が大部分を占めるという結果になりました。

アフターバーナーを筆頭に，どこまで票が伸びるか楽しみなところです。

- アフターバーナー（X68000）
- ファンタジーゾーン（X68000）
- サンダーフォースII（X68000）
- リボルティーII（X1）
- ボスコニアン（X68000）

スポーツ大賞

この賞は，いわゆるスポーツゲームの力作に贈られる賞です。昨年は，野球ゲームまっ盛りという感のあったこの部門ですが，今年はゴルフにビーチバレー，野球にエアホッケーとバランスよく分かれました。えっ，エアホッケーはスポーツじゃないって？　いえいえ，マウスをパドルに見立ててスコンスコンやるんだからヘタすりゃ全身運動モノでしょ，あーた。

- パワーリーグ（X68000）
- ダブルイーグル（X68000）
- シャッフルパック・カフェ（X68000）
- V'BALL（X68000）
- 野球道（X1）

海外移植ゲーム賞

日本のパソコンゲームもそれはそれで楽しいのですが，やはり海外にも良いゲームはたくさんあります。日本のものとは別に，発想の違いや観点の違いなどが逆に新鮮に見えたりもします。そういえば，あまりあちらのゲームに「お涙ちょうだい」ものはありませんね（関係ないか）。

そういった海外移植もののなかから，人気の高かったものをノミネートしてみました。

- Might & Magic II（X1/X68000）
- ローグアライアンス（X1/X68000）
- ウルティマ（X1）
- WINGS（X68000）

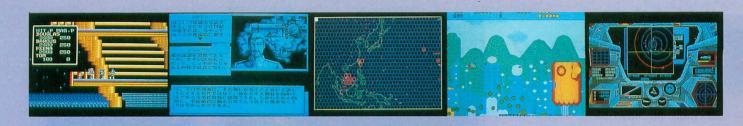
シミュレーションゲーム賞

今年はなぜかやたらとシミュレーションの秀作が多く，評価せざるを得なくなってしまったため，突如として設置してしまったこの部門。「三国志」や「SUPER大戦略」といったウォーシミュレーションだけでなく「プロダクション・マネージャー」といったタイプのシミュレーションが出たことも喜ばしいことです。じっくりと楽しみたいですね。

- SUPER大戦略（X1/X68000）
- 三国志（X1/X68000）
- A列車で行こうII（X68000）
- プロダクション・マネージャー（X68000）
- ウォーニング（X68000）

底抜け脱線ゲーム賞

読者の皆さんも一度は経験がある失敗買い。思っていたものとずいぶん違って悲しみにくれたこともあったでしょう。その思いをどうぞここにぶつけてください。ただし，もう怒りも治まって「あっはっは〜，あのときは呆然としたなあ」と笑って話せるようなものに限ります。推薦理由も一筆添えてくださいね。

ノミネートなし

ファンキーアイデア賞

この賞は，プレイヤーの脳天をぶち抜くような強烈なインパクトを与えてくれたゲーム，もしくは，いままで取り上げられなかったことを題材にしたゲームに対し，そのアイデアをたたえて贈られる賞です。

没個性のこの時代にあって，ポリシーを持って独自の世界を切り開いてくれたこの４作品を，今回はノミネートしたいと思います。

昨年の１位は，当然といおうかやはり「たんば」でした。大胆な発想とグラフィックがこの賞を確立したといってもよいでしょう。

- ねじ式（X68000）
- 琉球（X68000）
- ソフトでハードな物語２（X68000）
- デュアルターゲット 第4のユニット3（X68000）

最優秀パフォーマンス賞

今回惜しくもノミネートに漏れてしまった，もしくはノミネートまでに発売が間に合わなかった，そんな悔し涙を流したゲームもたくさんあったでしょう。でも，そんなゲームたちのなかにも「うっ，こ，これは凄い」とうならせた宣伝用ソフト，いわゆるデモソフトを作ってくれたものがありました。この賞は，パソコンショップなどの店頭デモで大活躍してくれたゲームに贈られる賞です。今回ノミネートされたこの３本，思わずお店で見入ってしまった方もいらっしゃるのでは？

- 夢幻戦士ヴァリスII（X68000）
- ミッド・ガルツゴールド68K（X68000）
- ナイトアームズ（X68000）

その他自由応募部門賞

この部門は，もうなんでもアリです。ここに載っていなくて，でもぜひみんなに知らせたいんだ〜っ，てことならなんでもOK。ただし，内輪ウケのものはちょっと，ね。昨年は「〜で賞」ってのがやたらと多かったです。今年はいったいどうなることやら……。こちらもスペースを空けて待ってますのでガンガン応募してください。

◆タイマーズのCDを買った。うーん，日本語でもこういう歌詞がのっかるんだ。平沢進の「時空の水」もよかったなあ。細野晴臣の「omni Sight Seeing」も気持ちよかったし。とてもTR-808使ってるとは思えないもんな。その辺のライブバンドよりずっと音がリアルだもん。映画もブラックレインなんてすごかったしな。神経衰弱ぎりぎりの女たちも楽しかったし。でも意表を突かれたのはパトレイバー（映画だよん）だな。思ったより面白かったしな。さすが微妙な暗さは押井守だ。今年は豊作だった，というんだろう。

で，肝心のゲームだけど。'88年に比べると揃ってたね。特に下半身（じゃなかった下半期）。上半期はアフターバーナーなんかのシューティング勢に押されていたけど，下半期になると変なゲームが現われたり（琉球，ねじ式），空前のゴルフゲームラッシュ（ジャックニクラウス，遥かなるオーガスタ，ダブルイーグルなど）など新しい動きが見られたから。T&Eよ，はやくX68000版オーガスタを作れよ（そーいえば，T&Eって私の実家から車で10分くらいのところにあるんだよね）。私おすすめのゲームはというと，有無をいわせず「ねじ式」だ。次点が「ファンタジーゾーン」と「ダブルイーグル」ってとこだな。頑張れ，ねじ式。

それにしても，ここ数年上位を独占したイース&ソーサリアン，大戦略，光栄シミュレーションの３大派閥は強いなあ。自民党もああなっちゃったことだし，新しい波が打ち寄せれば'90年代は楽しい日々を過ごせるかもしれない。

（荻窪　圭）

◆わがXシリーズにおいて，'89年の大きな事件は２つある。ひとつは，X68000専用の新作が登場し始めたことだ。いままでは，アーケードの移植，または既存ソフトのグレードアップバージョンがほとんどであった。それが，他機種にはない，X68000でしか遊べないゲームが登場し始めたことは，実に誇らしいことだ。

ソフトハウスにとって移植ものというのは，すでに他機種である程度の評判と儲けがあるため，１機種くらいコケてしまってもたいしたダメージにはならないが，新作でコケたらとんでもないことになる。他機種に移植してもヒットは期待できないし，ましてやX68000のように単一機種の性能をフルに生かしたものだと，移植さえもできないかもしれないのだ。それを考えるとたいした冒険行為だったといえるかもしれない。

こうした偉業を成し遂げたソフトハウス「アルシスソフト」「ズーム」「システムサコム」などには拍手を送らなければならないだろう。

さて，もうひとつの事件というのは，いわずと知れた'89年夏のビッグイベント「イース３」に，X1シリーズが参加できなかったことである。

これは，その後のX1シリーズの新作含有率にかなりの打撃を与えたようで，今後の行方が心配される。こうして見てくると，ソフトハウス側のX１からX68000への移行が確かなものになってきているようだ。

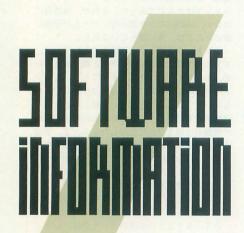
P.S.今年もおもしろいゲームがたくさん出ますように！　パンッパンッ　　（西川善司）

応募要項

1989 GAME OF THE YEARに応募されたい方は，今月号の愛読者カードの記入欄，または官製ハガキにOh!Xゲーム大賞とソフト名（今年は原則としてこれを明記することにします），そのほか自分の応募したい各賞の名前とソフト名，そしてそれらの推薦理由を明記して，Oh!X編集部までお送りください。

お送りいただいたハガキの中から，メッセージを採用させていただいた方に抽選で，任天堂の「ゲームボーイ」を１名の方に，アポロの「BATMOBILE」と，BNN発行の「RPG秘宝館」「RPG人名録」「RPG100の疑問」を各５名の方に，また100名の方にはOh!X４月号の表紙をあしらった特製「Oh!Xノート」をプレゼントします。締め切りは２月15日（当日消印有効）です。皆さんからのたくさんのご応募，編集部一同お待ちしております。

THE SOFTOUCH

SOFTWARE INFORMATION

X1/turbo
- ウルティマⅢ

X68000
- A-JAX
- モトス
- 信長の野望・戦国群雄伝
- Misty2
- ダンジョンマスター
- Musicstudio Mu-1

A-JAX
全8ステージのシューティングゲーム。2Dステージでは，超音速ヘリで真上からの視点で攻撃，3Dステージでは，ジェット戦闘機で真後ろからの視点で攻撃するというひと粒で2度おいしいゲームだ。

話題のソフトウェア

いや〜，先月はものすごいゲームラッシュで思わずページを拡張してお届けしましたが，今月はまた通常に戻して進めていきたいと思います。では，さっそく順を追って紹介していきましょう。まずはコナミから発売された**A-JAX**。2Dと3Dのステージで構成されたシューティングゲームです。なかなかの出来なので，ゲーセンで燃えた人もこれなら満足できるかな。同じくアーケードからの移植ゲーム，電波新聞社の**モトス**も発売されました。こちらはGAME REVIEWで紹介していますので，あとでじっくり読んでくださいね。そうそう，シンキングラビットの**倉庫番パーフェクト**もX1版とX68000版が同時に発売されました。こちらもGAME REVIEWで紹介しています。

先月紹介したカオスの**アルビオン**ですが，そろそろ発売ということで画面写真をお届けしましょう。発売が楽しみですね。

前作からわずか1カ月というのに，勢いづいてはやばやと第2弾が登場してしまったのがデータウエストの**Misty2**。基本的なシステムは前作と同じ。じっくり推理を巡ぐらしたい方にはおすすめです。あとシリーズものとしてはポニーキャニオンから**ウルティマⅢ**が登場。前作よりもはるかに操作性がよくなっています。ウルティマファンは見逃せませんね。

あと，シミュレーションゲームの大御所光栄からは待ちに待った**信長の野望・戦国群雄伝**がX68000に登場。光栄からはさらに**水滸伝**が'89年中に発売されます。お正月にじっくり家でやるのもいいのでは？　おっと，そうだ，もうひとつシミュレーションゲームがあったんだ。ボーステックから，あの**銀河英雄伝説**が出ます。早ければ1月ころには発売かも。乞うご期待！

そのほか入ってきた情報としては，M.N.M Softwareの X1用RPG**アルガーナ**，ポニーキャニオンから同じくX1用RPG**プール・オブ・レディアンス**，X68000用にはビクター音楽産業の**ダンジョンマスター**などが，それぞれ開発中とのこと。お年玉をためて楽しみに待っていましょう。では，また。

なんじゃ，こりゃー!

順位	タイトル	(前回順位)
1	アフターバーナー	1
2	テトリス	6
3	ソーサリアン	5
4	ジェノサイド	2
5	スタークルーザー	7
6	ファンタジーゾーン	3
7	ナイトアームズ	—
8	メタルサイト	—
9	V'BALL	—
10	サバッシュ	9

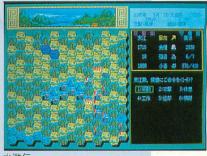
年も押し迫って寒くなってきましたが，皆さん元気に寝込んでますか？　浦川です。今月のTOP10ですが，しばらく続くと思われた均衡状態が突然バラバラになってしまいました。確かに「上位3作に喰い込めるソフトを期待する」とは言いましたけど，まさかこんな定番が入ってくるとは……ねぇ。よく見ると，ランクアップを果たしたソフトはX1にもあるものばかり。

なぜでしょう？　知ってるけど教えない。それでもアフターバーナーがこの大変動に耐え抜き，トップを堅持できたのは，X1ユーザーからの支持も厚かったのが大きいようです。

下位のほうには，年末の新作がぼつぼつ顔を

銀河英雄伝説

アルガーナ

水滸伝

新作ソフト情報

☆……12月1日現在発売中　★……近日発売予定
明記されたもの以外の価格については消費税は含まれておりません。

☆ウルティマⅢ

ミナクスとモンディンの滅亡後，平和だったブリタニアに次なる危機がおとずれた。海上には謎の島が姿を現し，山にはオークがはびこり始めた。海上の経済は完全にストップ，そして港にたどり着いた1隻の廃船に書きつけられた血文字……EXODUS。2人の魔術師に生み出された「何か」が復讐を開始せんとしているのだ！

ウルティマシリーズの3作目。今回はキャラクターデザインやシステムが一新され，心機一転の出来になっている。注目の1作。

X1/turbo用　5″2D版2枚組 8,800円
ポニーキャニオン　☎03(221)3161

☆A-JAX

2Dと3Dのステージで構成されたシューティングゲームで，赤い敵機を破壊して出てきたカプセルを取ることにより，パワーアップしていく仕組みになっている。また，赤い敵機の編隊を全滅させて出てきたカプセルを取ると，自機と同じ動きをするオプションが付く。ステージは8つ，もちろんジョイスティック対応だ。

X68000用　5″2HD版3枚組 8,800円
コナミ　☎03(264)5671

☆モトス

敵を体当たりではじき飛ばすという，単純明快なルールのアクションゲーム。こちらもアーケードからの移植だ。ラウンドは62，オプションはパワーパーツとジャンプパーツの2種。うまく使って攻撃を有利に運びたいものだ。MIDI対応というのもうれしい。

X68000用　5″2HD版 7,800円
電波新聞社　☎03(445)6111

★信長の野望・戦国群雄伝

いまや歴史物SLGとして名高い「信長の野望」シリーズ最新版がいよいよX68000にもリリースされる。基本的な路線は同じだが，今回は新たに人材登用という要素を加えたのが大きな特徴。個性あふれる武将は200名を超え，これらの人材を取り立てて内政のサポート役としたり，領地を委任したりできるのだ。また，戦争シーンのルールも若干システムアップされ，夜戦や籠城戦などの戦略が新たに加わった。

X68000用　5″2HD版2枚組 9,800円
光栄　☎044(61)6861

☆Misty2

早くもシリーズ第2弾が登場。今回はユーザーから寄せられたものも含め，5つのシナリオが用意されている。聞き込んだ情報から推理し，質問に答えるというパターンは前作とまったく同じ。紙と鉛筆を用意して，得た情報をじっくり考えよう。お便りコーナーや探偵講座もあるぞ。

X68000用　5″2HD版 5,000円
データウエスト　☎06(968)1236

★ダンジョンマスター

欧米で大人気だったアクションRPGが，FTL社自身の移植でビクター音産より発売になる。ゲームは破壊された世界を復元するカギとなるファイアスタッフを手に入れるため，4人パーティで全14レベルの迷宮に挑むというもの。

落ちているものを見る，食べる，どこかに置くといった幅広い行動ができ，メニューを開いている最中でも敵は自分の判断に基いて動いているという凝りようだ。武器のリーチや，攻撃の方向によるダメージの違いにまでこだわり，あたかも自分が実物のダンジョンに入っているかのような感覚が味わえる。

X68000用　5″2HD版 9,800円
ビクター音楽産業　☎03(423)7901

★Musicstudio Mu-1

Musicstudioからリアルタイム録音機能を除いて，細かな改良を加えたMIDIプレイヤー。PC-9801用の「ミュージ君」,「ミュージ郎」のデータがコンバート，演奏できるのが最大の特徴だ。

ミュージックシェルも大幅に機能アップし，操作性も向上。そのほか，カラー化されたウィンドウにマジックパレットで描いた絵が読み込めたり，ウィンドウ内にテレビが映るイメージモードも付いた。サイバースティックにも対応する。

X68000用　5″2HD版　価格未定
サン・ミュージカル・サービス　☎03(419)8839

出してきました。特に票の伸びが期待できそうなのは，ナイトアームズとメタルサイトの2本。どちらもみんなが葉書を出した頃はデモしかなかったはずなんですが，そのデモのインパクトがすごかったということでしょう。発売後の動きに注目だぁ。

この先，モトスにラグーンにスーパーハングオンにサンダーブレードにA-JAXと，強豪がぞくぞくと控えています。しばらくGAME OF THE YEARとの兼ね合いもあって，得票は乱れそうですが，その混乱の中から抜け出るのはどこか楽しみですね。GAME OF THE YEARのほうも，おハガキお待ちしてます。ではまた。

プール・オブ・レディアンス

アルビオン

信長の野望・戦国群雄伝

ダンジョンマスター

GAME REVIEW

GAME REVIEW

今月は年末年始のせいもあってか，紹介したいゲームが7本も出てきてしまいました。X1には「ヒーロー・オブ・ランス」と「ウルティマIII」の2本を，X68000には「シャッフルパック・カフェ」「フラッピー2」をはじめ5本を用意しました。

ヒーロー・オブ・ランス

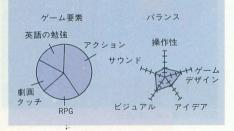
言わずと知れた海外の移植ゲーム。RPGと思いきや，8人の冒険者をそのつど替えて剣や魔法で戦っていくアクションゲームだ。

▶あなたは「ランス（昔のカープの外人ではない）の仲間たち」となってドラコニアンを倒し，「ランスのヒーロー」となるのが使命だ。このゲームのマニュアルには，大胆にも「これはRPGではない」と書かれているが，どう見たってアクションRPGだろう。「仲間たち」といったって，同時に操れるのはひとりだから，大昔にやった某レリクスのヒョコヒョコ歩きを思い出してしまうのだった（もっともこっちはセーブできるが……）。

RPGではないというだけあって，なかなかアクション性の強いゲームになっている。特に，敵がヒョコヒョコと歩いてきて剣をブンブン振って，こっちもブンブン振っていると，そのうち敵がぱん！　とはじけるようにやられるのが小気味よい。なんといっても，英語モードでやれば語学力がつくのがよい。さすが移植ソフトである。ちなみにこれは，ウルティマを移植したポニーキャニオンの移植である。

熱中度▶▶▶▷▷▷▷　（亀）

▶皆さんご存じのテーブルトークRPG，AD&Dのコンピュータゲームです。ドラゴンランス戦記は小説化されているので，目を通したことのある人なら「ゲームの遊び方」を読めば，すぐに遊べるでしょう。

このゲームは意外（失礼！）にもアクションゲームなんですね。てっきりフツーのRPGだと思っていた私は，ちょっと新鮮な

驚きを覚えてしまいました。基本は8人のパーティのなかから状況に応じてひとりを選び剣や弓，魔法などで戦っていくというもの。でも，アクションゲームなのであまり魔法を多用しないほうが楽しめます。それに，なかにはあまり使えない魔法もあるので，ひととおり試してみるといいかもね。

で，肝心の操作性なんだけど，ちょっと「う～ん」とうなってしまう。アクションが主体でありながら，スピードがいまいち遅いんです。もっとバタバタと敵を倒せたらよかったのに。原作がいいだけにちょっと残念。次回作を期待しています。

熱中度▶▷▷▷▷▷▷　(K. M.)

X1turbo用　5″2D版4枚組　7,500円（税別）
ポニーキャニオン　☎03(221)3151

ウルティマIII

一連のウルティマシリーズからの移植第4作目。パーティ編成は4人まで。前作よりも操作性がよくなっている。

▶これは違う！　確かに，前作とはそんなに違わないかもしれない。そんなに速くなったわけでもない。しかし，そのキャラクタや，ダンジョン内の壁には気合いが感じられるのである。ソーサリアンみたいなキャラクタになったかもしれないし，なんとか楽しめるほどのスピードになったと思う。それもこれもこの気合いが入った，ということのせいなのである。

ウルティマのIとIIで，悪のモンデインとミナクスは倒された。しかし，この2人の間にはなんらかの契約があり，不気味ななにかが生み出されたようである。それを確かめるのがそなたの天命なのである。いままでウルティマをやったことがない（やりたくなかった？）人も，この4作目は一見の価値があるはず。特に，このダンジョンは進化して，壁もドアもきれいになって，やっと人並になったと言えるのである。よ

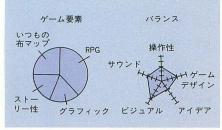
かった，よかった。ストーリーはもともといいから面白いかもよ……。

熱中度▶▶▶▶▷▷▷ (鶴)

▶あの名作ウルティマシリーズのパート3が登場です。ゲームシステム自体はあまり変わっていませんが，前回までのI，II，IVで問題だった操作性の悪さが，かなり改善されています。これにより，いままで面倒だったマップの移動もスムーズになり，イライラすることなくゲームを楽しめます。

今回も，ロード・ブリティッシュに召集された勇者たちの冒険の物語で，この広い！　ひろ〜い世界を馬や舟に乗りながら旅をします。

慣れている人はともかく，初めてこのゲームをプレイする人は，ずいぶん戸惑うと思いますが，この寒い時期，部屋にこもってゆっくりウルティマの世界を旅してみるのもいいんじゃないかな。いまさらウルティマなんて，と言わずに一度はまってみてはいかがでしょうか。ポニーキャニオンさん，この調子で次に予定されているウルティマVのほうもしっかり移植してもらえるとうれしいな。

熱中度▶▶▶▶▷▷▷ (純)

X1turbo用　5″2D版2枚組　8,800円(税別)
ポニーキャニオン　☎03(221)3151

シャッフルパック・カフェ

マウスをパドルに見立ててパックを打ち合うエアホッケーゲーム。対戦する相手は，10人の宇宙人と1体のロボットだ。

▶デパートの屋上なんかに置いてあるエアホッケーみたいなゲームだなと思ったら，冗談でなくそうだった。またの名を「疑似3Dのひとりテニス」とも言う。このテのゲームはテンキーの4，6でパドルの移動を行うものと相場が決まっていましたが，そこは我らがX68000ですよぉ。標準装備のマウスを使って前後左右にパドルをガンガン動かすことができちゃうのだ。これはもうマウスがパドルになってしまったようなもので，机の上の物を片づけておかないと，パニックになってしまう。間違っても飲みかけの缶コーラなんか置いとかないようにね。でないと机の上にコーラの海ができてしまいますぞ。

コンピュータと試合をするんだけど，強い相手になると自分の打った玉（？）をほとんど打ち返してくるので，まるで鏡に映った自分を相手にしているよう。腕がだるくなって，マウスが汗まみれになるほど熱中してしまうゲームだ。

熱中度▶▶▶▶▶▷▷ (H. K.)

▶2年以上酷使に耐え続けた我がマウスには苛酷過ぎる試練であった。ローラーにたまったわずかなほこりと，ボールの些細な傷が微妙に手元を狂わす。地球人らしい洗濯屋を相手にしているうちはまだ楽勝だった。問題は，奇妙な笑い声を出し，少しでも負けると怒り出す短気な宇宙人たちだ。トーナメントする前に，酒場にたまったヤツらから適当に相手を選んでクリックする。プロフィールを見てびびりつつも，これもまた流れ者の宿命。おっと，カウンターの陰に隠れているロボット野郎が弱そうだ。一夜の慰みを求めるならそれでもいい。負ければ負けたの風が吹く。だが，トーナメントともなると話が違う。わかったのは，新しいマウス（そしてマウスパッド）を手に入れてから再来すべきこと。おお，あの超能力を使うキタない女王め！

ただのエアホッケーがここまでエンターテイメントになるとは。あとは元気なマウスさえあればナイス。

熱中度▶▶▶▶▶▷▷ (K)

X68000用　5″2HD版　7,800円(税別)
ブロダーバンドジャパン　☎03(341)1135

セリフももっとね!

この前，テレビでバッテン荒川（知らないですか？　九州のローカル芸人なんですけど）の24時間密着取材っていうのを見ました。このバッテンさんの芸が面白いんですよこれが。どーゆー芸かというとおバアさんのかっこう（本当は40歳ぐらいのおっさんなんだけど）をして早口の博多弁でしゃべくるわけです。これが実際に見ると面白いんだ。私は方言（この場合は博多弁）っていうのもののインパクトの凄さを再認識してしまったわけです。

で，私は思ったのですが，なぜアドベンチャーゲームのキャラクタに全然方言を喋らせないんですか，シナリオ屋さん。たとえば漫画なんかでも特徴的なキャラを作るのによくしゃべらすんですよ。たとえば「〜だっちゃ」とか。うる星やつらのラムちゃんだってすぐわかるでしょ。そのぐらいインパクトあるんですよ。だから1回やってみてくださいよ，だまされたと思って（だまされたりして）。

それにしても，またバッテン荒川が見たいなあ。ビデオでも出てないかなあ。それとも九州まで見に行っちゃおうかなー。 (で)

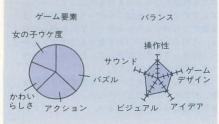
フラッピー2

ブルーストーンを敵をかわしつつブルーエリアに運んでいくアクションパズルゲーム。キャラクタの動きがかわいらしい。

▶デービーソフトと言えばフラッピーと言うほどの名作，アクションパズルゲームの大御所が帰ってきました。その名もフラッピー2。「キング・フラッピーがあったにもかかわらず，フラッピー2とはこれいかに？」「1，2がなくてもラブ3(スリー)と言うが如し」う～む，奥が深い。このゲームは6年ぶりの復活ということもあって，随分パワーアップされているようです。ただ，仕掛けもキャラも，同じくデービーのラプテックに通ずるものがあり，目新しさはありません。ゲーム自体も私がフラッピー通だったためか，サンプル版だったためか，さほどムズくはありません。しかし，もともとのゲーム性の高さからでしょうか，キャラたちのかわいさも手伝って，女の子なんかとワイワイ遊ぶにはとっても有効であると思えます（試した人は報告してね）。

スーパーアクションゲームで反射神経ばかり発達してしまったX68000ユーザーには調度よい脳みそ刺激剤になるでしょう。

熱中度▶▶▶▶▷▷▷　(S. K.)

▶昔遊んだ，あのフラッピーが今，フラッピー2となってX68000にやってきました。このフラッピー2，前作のフラッピーとルールなどは基本的には変わっていない（つまり，“半ずらし”なんかも使うわけですね）んですが，そこはやっぱりニューバージョン。主人公のフラッピーくんはテヤテヤ光

っちゃってるし，背景はきれいだし，地面は斜め向いてるしで（別に前作が悪かったわけじゃないけど）いろいろとよくなっています。“5”のキーを押さなくてもちゃんとキャラクタが止まるとはねぇ(そりゃFM版だけか)。いい時代になったよなぁ，としみじみ思ってしまうのでありました。

うーん，それにしてもほのぼのとしたゲームだ。なーんか，お花畑はきれいだし，キャラクタの動きはむにゅむにゅだし，岩はサイコロキャラメルの空き箱みたいにころころ転がってくし，敵キャラのエビーラはレッドロブスターにいそうだし。うーん，メルヘン。おすすめの1本です。

熱中度▶▶▶▶▶▶▷　(で)

X68000用　5″2HD版　8,800円(税別)
デービーソフト　☎011(807)6700

V'BALL

「熱血高校ドッジボール部」で燃えた人にはうれしいビーチバレーゲームだ。操作は比較的やさしく，誰にでも楽しめる。

▶熱血高校ドッジボールに続く球技シリーズの第2弾がこれ，見てのとおりのビーチバレーです。♪苦しくたって，悲しくたって……，で始まるテレビアニメは室内バレーボールだったけど，ビーチバレーは浜辺で1チーム2人で戦うのです。操作方法は矢印カーソルの指している人間を動かして，Aボタンでジャンプ，Bボタンでレシーブ，サーブ，スパイクとすっきりまとめられているし，面倒臭いトスは相棒が勝手に上げてくれるので，初めてでも安心なんです。

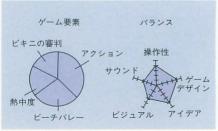
まさかと思うけど，このゲームをただのビーチバレーだと思っている人がいたら，意識の改革が必要だ。レバーとボタンの組み合わせでいろいろな必殺技が使えるのだ。「いろいろ」と書いたけど，マニュアルにはひとつしか紹介されていない。探せばきっとたくさんあると思うんだけど，どうなんでしょう。8等身まではいかないが，キャラクタの動きはなかなかリアルです。

熱中度▶▶▶▶▶▷▷　(H.K.)

▶やっぱスポーツっていいよなぁ。理屈抜きで楽しいもんなぁ。ゴルフも野球も捨てがたいけど，ビーチバレーってのもいいよなぁ。いかにも「夏」してるし。冬にこそこたつ入ってアイスクリームがお洒落な時代だもん，発売時期はよっぽどねらってたんだろうなぁ。○○○○○ド買って視力落として，反射神経削るよかよっぽど楽しいよなぁ。特に年末年始なんかは友達が集まる機会も多いから，2人で同時にプレイできるのは絶対においしいよなぁ。別に友情にヒビが入るなんてこともないだろうし。

僕はひとり寂しく遊んでたせいか，スポーツゲームが得意なせいか，初めてこのゲームを触ってから2時間でメジャーサーキットを制覇しちゃったんだけど，まさか最終ステージがあんなところとはねぇ。審判のおねえさんもついてきてたのはビックリしたけど，めでたくテスタロッサを手に入れたのさ。F14が欲しかった僕だった。

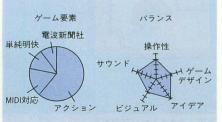
熱中度▶▶▶▶▶▶▷　(S. K.)

X68000用　5″2HD版　7,900円(税別)
シャープ　☎03(260)1161

モトス

ソーラー・ベースに巣くう敵「スペースビー」を自機「モータースパナー」の体当たりでベースからはじき出すというゲーム。

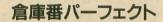
▶「モトス」は，1985年にナムコから発表されたアーケードゲームです。自機を操り，敵機を体当たりではじき飛ばし，フィールド上から下へ落とす，単純ながらも一風変わったゲーム内容。皆さんが気にする移植の出来ですが，私が見た限りでは敵の動きはまずまずで，かなりいい線行ってるとは思うんですが，このモトス，アーケードでは縦画面だったんですよね。「ドラスピ」みたいにディスプレイを縦に置いてプレイできるモードがほしかった。音楽にはかなりの力が入っていまして，話によればディスクの半分以上を音楽関係が占めているとか。なんと今回はMIDI楽器にも対応していて(『MIDI対応説明書』というマニュアルまで付いてくる！）対応楽器は，いまや音源モジュールの定番と化したMT-32と，シンセ界のX68000ことM1。作曲者は一連の移植ものの音楽を担当してきたYu-You氏。ゲームミュージックマニアは見逃せない1本かも。

熱中度▶▶▶▶▶▷▷ (善)

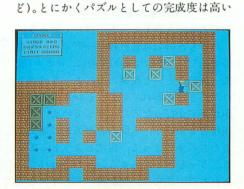
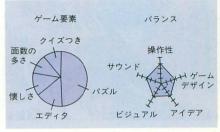
▶ゴチごちゴチゴチ。ごチン，ごちごち。うーん，敵をぶつけて落とすだけという単純なルールだけど，それだけに燃える！重たい敵にゴチンとはじかれたときには結構ムッとくる。それに，最近では珍しいことにパターン性がない。純粋にやる人のウデで決まるというのが好きだなぁ。アイテムの選択によって作戦が全然違ってくるから，上手くなったからって簡単に飽きてしまうゲームではなさそうだし。ゲームセンターでやってたときはパッとしないと思ったが，実は家庭向きの“隠れた名作”だったのだ。

移植はさすが電波だけあって文句のつけようのない出来。今度はディスプレイを縦にしなくても縦長の画面が楽しめるようになったのもうれしい。欠点のない出来であります。しかし，その代わりウリもあんまりないのが心配だな。いいゲームだから売れてほしいんだけどなぁ。

熱中度▶▶▶▶▶▷▷ (H. U.)

X68000用　5″2HD版　7,800円(税別)
電波新聞社　☎03(445)6111

倉庫番パーフェクト

昔懐かし，あの「倉庫番」の登場です。X1も同時に発売中ですが，今回はX68000のほうを紹介します。

▶倉庫番です。68版になっても（見た目は少々ハデになっていますが）シンプルで深い味わいは変わっていません。ハマります。

要するにパズルで，倉庫の中の荷物を指定された場所に揃えるというものなのですが，これが曲者です。なんと「荷物は押せても引っ張れない」のです。このルールのおかげで何回「おーっ」という目に遭ったことか（それで結構意地になるんですけど)。とにかくパズルとしての完成度は高いのでかなり楽しめる（苦しめる？）と思います。操作性も上々ですし（ただゲーム中スコアウィンドウが移動する際一瞬自分の動きが止まるのが気になりましたが)。

当然面エディタやすでに解いたステージを再現するトレースなどありそうな機能もしっかりあります。あとこの全部の面の中に「絶対解けない面」が1面だけ入っていてまたそれが狂おしいまでにいい味を出しています。お正月にお勧めの1本でしょう。

熱中度▶▶▶▶▶▷▷ (哲)

▶フラッピーが「フラッピー2」となってX68000用にやってきたように，倉庫番も「倉庫番パーフェクト」となってX68000にやってきました（ああ，そういえば昔このシリーズの「TNT BANG-BANG」が4面までしか解けなかったんだよなー)。このX68000版倉庫番は面数306，そしてパズルゲームということで，問題を自分で作れるエディタ付きです。また問題も，1問1問が結構ムズい。長く楽しめそうです……。

しかしですね，このパズルゲームっていうのは，同じようなパズルが延々と続いてるだけ，っていうんでいいんでしょうかねー，私はそう思います。本当に306面全部ユーザーに解かせるつもりなら，何かイベントがあってもよかったんじゃないですかねー。どこまで進んでも，問題が難しくなっていくだけじゃねー。'90年代に向けて，これはっ，ていう工夫がほしかった気がします。今一歩だったなー。

熱中度▶▶▷▷▷▷▷ (で)

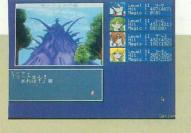
X68000用　5″2HD版　6,800円(税別)
シンキングラビット　☎0797(73)3113

THE SOFTOUCH

●レナム

AVG要素を含んだマウス操作のRPG

Kunitsu Yoshio
国津 良男

新規参入会社ヘルツの第1弾。街や村の中，そして各イベントはAVG方式，ダンジョンの中や移動，戦闘シーンはRPG方式というゲームだ。邪教神バーディスを倒すことがこのゲームの目的。

X68000用 5"2HD版6枚組 9,800円（税別）
ヘルツエンジニアリング ☎03(371)3081

編集部の片隅にある棚には，多くのゲームソフトが積まれている。以前はちょくちょく拝借しては遊んでいたのだが，最近はGAME REVIEW用以外にまで進んで手を出すことは少なくなった。それでも，話題についていけなくなったらヤダ，という不純な動機から遊ぶことはある。しかしそのような場合は，数十分遊んでみて，

ん，だいたいわかった

で，終わりである。似たようなソフトが多いので，（一時的にしろ）ゲームに飽きてしまったのだ，と言ってもいい。

何かの本に，こんなことが書いてあった。「車のセールスマンは，車が売れなくなるように働いているのだ。なぜならその行動は，需要者を減らしていることにほかならないから」

私たちは，ゲームに飽きるために，ゲームをしているのかもしれない。なぜならその行動は，ゲームに対する許容量を減らしていることにほかならないから。

「運命」だって聴き飽きる

と，いきなり消極的な書き出しをしてしまったけど，SPECIAL REVIEWはきっちりといきましょう。

AVRPGってなんだろ？

レナムのジャンルはAVRPGということになっている。言うまでもなく，AVはアドベンチャー，RPはロールプレイングの略である。つまりこれは，アドベンチャーとロールプレイングの融合体ってわけだ。

AVGかRPGかを見分ける方法のひとつとして，キャラクタに自分の好きな名前を付けられるかどうかで判別する，ってのが考えられるけど（ウルティマは違うな），このゲームの場合は不可，だ。主人公はティアとリーク，以下キャラクタの名前は，あらかじめ定められている。あの面倒で楽しいキャラクタメイキングは必要ない。

これらの出来上がった（酔っているわけではない）キャラクタを引き連れて，何をするかっていうと，結局はやっぱり，

「わるもんをやっつける」

のである。もうちょっと，詳しく話そう。

わるもんの名は邪教神バーディス。邪悪の元凶だ。こやつは1500年前に，レナム（魔法一族のこと）と聖剣の力によって滅ぼされた。と，思われていた。しかし，実は封印が不完全だったのだ。そしていま，邪教四神官とかいう，人間の力を遙かに超えた者たちの手によって，復活しようとしている。うーむ困ったちゃん。っつーわけで，われわれの使命は，邪教神バーディスを完全に滅ぼすことにある。そのためには，どこかにいるレナムと，どこかにある聖剣を捜し出さなければならない。では，邪神教界や悪魔界の妨害に負けずに，頑張ってくれたまえ。

とまあ，そういうストーリーだ。おお，いかにもゲームのストーリーではないか，という感じは，ある。じゃ，そのさわりを見てみよう。

冒険はこーして進む

ゲームは，ある平和な街の，ある通りから始まる。最初の主人公は，魔法使いの女の子ティア。かわいいときれいの中間タイプだ。彼女こそが，邪教神バーディスを倒す旅をしている張本人である。まずは，この街の人々に話を聞いてみよう。街の中はアドベンチャー方式になっているので，「聞く」をクリックすればいい。

＊　　＊

ティア「すみません。あの，今日は何かお祭りでもあるのですか？」
街人「あ？　お前さん何も知らないのかい？　今日は年に一度の剣術大会の日なんだぜ」
ティア「剣術大会ってどこでやっているの？」
街人「この先だよ」

＊　　＊

というわけで，大会を見に行ってみる。同様に「移動」をクリックすればいい。ドラクエみたくトコトコ歩いていく必要はない。もし，街の人に話を聞く前に移動したりしようとすると，「その前に，ちょっと辺りを見回してみましょう」，なんて表示される。親切設計といおうか，一本道といおうか。どっちにしても，アドベンチャーというわりには，ごく簡単なものなのだ。

剣術大会の会場では，優勝者のリーク（剣士・男）と出会い，以後，彼と一緒に旅をすることになる。というか，ここからリークが主人公になる。いきさつを抜粋すると，こうだ。

＊　　＊

宿屋に泊まるとセーブできる

ティア「ねぇ，あなたリークっていうんでしょ？」
リーク「えっ？　ああ，そうだけど。何？」
ティア「私の名前はティア。ティア・ネイ・ラーパ」
リーク「んん！　わかった！　オレのサイン欲しいんだろう！　フッ……もてる男はつらいぜ」
ティア「……？　お願い，私と一緒に聖剣とレナムを捜して!!」
リーク「なぜ俺のような男なんかに？」
ティア「……だって，私……。聖剣使えないもん」
リーク「あら，まぁ……」

＊　　　＊

とんちんかんな会話だこと。２人とも，

明るくて，どっかヌケている

といったタイプの性格なのだ。

軽いタッチの作品には，このテの人物が頻出するね。あかねちゃんも，魔美ちゃんも，まるちゃんも，そして山瀬まみも，ゆうゆも，カケフ君も，みーんなこのタイプだ。しかし，なんといっても代表的なのは，

サザエさん一家

だろう。それは置いといて，と。

２人は情報を求めてミーマの村に行くことになる。その前に，この街で武器や薬を買っておこう。特に，街（村）から街への移動は危険だから，ヒットポイント回復の薬と，マジックポイント回復の薬はたくさん買っておきたい。私はここでケチってしまって，何度となくやりなおすハメに陥ってしまった。街から街への移動も，移動先をクリックするだけ。広域マップが現れて，勝手に進んでくれるので楽だ。

で，やっぱり，道中でモンスターが襲ってきたりする。これらのモンスターの絵は，とても強そうに，かつ気持ち悪く描けていてイイね。それに比して，人物の絵柄は，同人誌っぽくって，かつファンタジー漫画っぽくて（まったく個人的な趣味で）好きじゃない。バスタードの類は受け付けない体質なのだ。(て)さんはうる星やつらとバスタードを両立させているみたいだけど。

モンスターとの戦闘は，ドラクエ方式とたいして変わらない。こっちが目標と戦闘方法を指定してバン！　次は敵の番でバン！（だじゃれじゃないよ）ってやつ。モンスターと戦って勝つと，経験値が上がっていくのはもちろんだ。経験値が上がるとレベルが上がる。レベルが上がると魔法使いのティアは魔法を憶えていく。と，こういう仕掛けなんだな。

運よくHPがなくならずに（なくなると，

森までくれば城まではあと少し

中ボスといったところか

即ゲームオーバー)，ミーマの村に到着すればラッキー。体力はそこの宿屋で一気に回復できる。この村での展開はこんな感じ。

＊　　　＊

リーク「ねぇ！　おねーさん，聖剣とかレナムって知ってる？」
女の人「ん……そういうことなら，教会に行けば何かわかるかもしれないわ」
リーク「教会かあ……どうも!!」

＊　　　＊

そんなわけで，ある程度軽い，しかし本筋はシリアスなお話が繰り広げられていくわけだ。のちには，４人パーティになったり，ダンジョンが待っていたりする。森の中なんかは，４次元迷路になっているし，終盤には紙芝居もありと，単調にならないように工夫しようとしている。

ときに，ティアとリークが愛だの恋だのというよーな仲になっていくことは，容易に想像できるが，まさにその王道から一歩も外れずに進んでいくようすは，なかなか恥ずかしいものがあった。お前ひがみじゃないのか，という噂もあることはある。

さて，総合得点は？

遊んでいてうれしかったのが，随所に見られる，ちょっとした心配りだ。すこしでもいいものを作ろうとする，心意気みたいなものが感じられる。

たとえばセーブ。８カ所までのセーブができるようになっていて，各々が，何月何日何時何分にセーブされたかの情報を持っている。だもんで，あれ，最後に遊んだデータは２番だったっけか，３番だったっけか，などと迷う心配がなくなった。

それから，BGM。AVとはアドベンチャーのことだと言ったが，このゲームの場合，オーディオ・ビジュアルの意味にも取れる。とくにオーディオだ。MIDIを介して，ローランドの音源モジュールMT-32とつなぐことができる。それはもう絶句モノの素晴らしい音楽がジャンジャカ鳴る。本体の音源だけでも結構いい音してますけどね。

また，マウスカーソルの動きも凝っている。２番目のウィンドウを開くと，カーソルはその頭を指してくれる。閉じると，前のウィンドウのもとのカーソル位置まで戻ってくれる。小さな親切，大きなお世話って人もいるだろうけど。

そんでもって，文字の表示もそれなりに速い。私は，１文字ずつトロトロ表示していく方式には耐えられない。何が楽しくて，遅延タイムを設けているのだろうか，といつも思う。マウスを押すと，素早く表示するような仕様のゲームだと（このゲームもそうだけど），私のマウスは，

始終カチカチ鳴りっぱなし

になってしまうのである。

とまあ，細かい点での気配りはできていると思う。だけど，ゲームのシステム自体は，AVRPGなんて言ってるけど，特に目新しさは感じなかったんだな。もし私が，このSPECIAL REVIEWの依頼を受けなかったとしたらどうだろう。編集部の棚にひっそりと置かれた「レナム」に，進んで手を出して遊んでいたかどうかは，疑問だ。

でもまあ，奇をてらったものが高い評価を受けるのにふさわしい，ってわけでもないし，普通のAVGやRPGで遊びたいって人もたくさんいるだろう。いまいちセンスが感じられないけど，全体的な完成度は決して悪くはないんだから，

総合得点75点

ということにしておこう。

おわりに

信じられないことに，1989年も，もうすぐ終わり。去年はゲーム三昧の年末を送ってしまったけど，今年は音楽でも聴いて過ごすことになるのかな。12月25日には，「小川範子の単行本」予約特典，

範ちゃんのクリスマスカード

が届くことになっている。楽しみ楽しみ。なんたって私は，小川範子ファンクラブ会員ナンバー06039だからね。おまけにクリスマスは，私の誕生日でもあるのさ。

●メタルサイト

エンディングまであと何千里?

Nishikawa Zenji
西川 善司

最近「難しいシューティングが少ない」とお嘆きの貴兄に，ぜひプレイしてもらいたい1作です。尋常でない敵の数，その上ナパーム弾の補給もほとんどないのです。腕試しにはちょうどいいかも。

X68000用　5"2HD版5枚組　9,800円(税別)
システムサコム　☎03(635)7609

どーも，善司です。あのシステムサコムから3Dアクションゲームが登場です。新しいパソコンユーザーには,「ノベルウェアばっかし出してるアドベンチャー専門のソフトハウスじゃーん」っていうイメージがあるかもしれませんが，実はアクションゲームはサコムの得意分野なのです。古いところでは，3Dワイヤーフレームの「バリアント」がありますし，完全3Dの物体がウニョウニョ動きまわる「ゾーン」，X68000用「フルスロットル」顔負け(!?)のドライブゲーム「ハイウェイ・スター」などなど。サコムはこれらのゲームをすべてPC-9801専用に出しています。もしPC-9801かPC-286を持っていたら，ぜひ前述したソフトをプレイしてみてください。

BGMも聴き逃せない

メタルサイトはBGMがMIDI対応で，MIDIボードとMT-32があればゴージャスで美しいサウンドが楽しめます（ミュージックモードもある！）。最近MIDI対応のゲームソフトがぼちぼち出始めてきたことは，ゲームミュージックファンとしてはうれしいですね。さて，このメタルサイトのBGMを作曲した安芸出（あき・いずる）氏は，名前はそれほど前面には出ていませんが，アーケードゲームでは「ワンダー○モ」，パソコン方面では「ソーサ○ア○」「イー○3」，あとアルシスの新作「ナイト○ーム○」，ヘルツのRPG「○ナム」などなど，数多くの名曲を生んだスーパーコンポーザーなのです。メタルサイトでは，各ステージごとにフュージョンを中心とした静かで「宇宙物」っぽい（?）BGMを聴かせてくれます。私のお気に入りはステージ2の曲で，なんとなくラテンの香りのする曲調に感動してしまいました。彼の今後の活躍にも注目したいですね。

ミッション・メタルサイト

「AD.2116年，E.L.M.（地球開放軍）は過激化するU.N.（国際統合連盟軍）との戦いに終決を迎えるべく最新の機動兵器を敵最終兵器破壊のために単身出撃させた」。「メタルサイト」とは，この作戦のコードネームなわけです。プレイヤーは3人のパイロットのなかからひとりを選び，まず入隊試験を受けなければなりません。これは，実は敵基地内部を想定したシミュレーションで，ゲームの1面目に当たります。

使用できる武器はバルカン砲とナパーム弾の2種類。バルカンの弾数は無制限。ナパーム弾はプレイヤーと同じ水平軸にいる敵を全滅させる強力な武器ですが，数に限りがあります。画面左下のゲージがその数を表しています。基本的には，ザコキャラを破壊しつつ次のエリアまでたどり着くことが目的で，シールドレッドの状態で敵の攻撃を受けるとゲームオーバーとなります。また，ナパーム弾の使用を2ゲージ内で抑えれば，ステージクリア時にシールドが1段階リペアされます。あと特に意味はありませんが，このとき撃墜した敵の数に応じた階級章が与えられます。

ステージ1

敵要塞内部の情報をもとに作られたシミュレーションで，ミッション・メタルサイトへ参加するための試験でもあります。このステージは1面ながらも結構難しいのです。ゲーム開始直後，ジョイスティックのボタンを1回押して，どっちのボタンがバルカンかナパームかを確認して，バルカンのボタンは押しっ放しに。にわかに小回りを始め背景の壁にぶち当たらないように敵撃墜の準備をします。

うわーっと，人工衛星みたいのが目の前に現れた！　のわーっ，敵のミサイルに当たるっ……　あれ？　当たらないぞ。そう，そうなんです。1面の敵はミサイルをわけのわからん方向に撃ってくるのです。ですから，自機を置く位置を決めてそこを中心にミサイルを避ければ，当たることはほとんどなくなるはずです。障害物の配置さえ頭に叩き込んでしまえば，楽勝楽勝。ナパームは全然使う必要ないです，はい。

ステージ2

んーっ。このエスニックなパーカッション，いーなぁ。さてさて，2面の舞台はライル草原地帯。初めての実戦です。前ステージでは敵は人工衛星野郎ばかりでしたが，さすがは実戦，敵も自機と同じような機動兵器が次々と出てきます。このステージからは敵ミサイルは自機めがけて飛んできますから，1カ所に止まっているとあっという間にゲームオーバーになってしまいますよ。電波の「スペースハリアー」をプレイしているかのごとく，円運動をしながら攻撃しましょう。スペハリのときより小さな円を描くのがコツです。あと，レバーを下に入れっ放しにしても地面に墜落することはないし，障害物の木も破壊可能ですので画面下部も有効に使いましょう。

ステージ最初と中盤で，巨大ロボットが出てきたら自機の移動を大回りから小回りへ切り替えましょう。大きさに圧倒されて

ナパームを連射しがちですが，なるべくバルカンで倒しましょうね。どうでもいいことですが背景キャラのシマウマやキリンに混じってコイサンマンがいます。探してみよう。ところで，このシマウマとかキリン，ただボーッとつっ立っているだけじゃなくて，群れをなして画面を走り過ぎてったらもっとかっこよかったのに。

ステージ3

ステージ3は，グラデーションの美しい空の広がるフゴ砂漠地帯。自機を下に持っていくとわかりますが，ほら，おぉー，砂塵が巻き上がりますね。すごいすごい。のわっ！　低空飛行していたらダメージを受けました。なんと今度の岩や石柱は破壊不能みたいです。ということは画面下部に行くときはかなりの注意が必要みたい。

白くて肩のでかい機動兵器の編隊攻撃はナパームで消しましょう。ステージ最後の巨大装甲車も，自機を大きく左右に振りながらナパームで攻撃しましょう。ナパームは画面いっぱいに広がっていくので，左右の端から撃っても当たります。この性質を有効に利用しましょう。

ステージ4

ステージ4は，つらら状の鍾乳石の広がる暗い洞窟です。この鍾乳石が上下の天井からビッシリ伸びているのを見て圧倒されがちですが，これはバルカンで破壊可能なのです。敵機は1面でお世話になった人工衛星野郎ばかりで，相変わらずとんでもない方向へ撃ってくるだけ。画面下部に自機を落ち着けバルカンを連射していればOK（1面の攻略法がそのまま使える）。ナパームは1面同様使う必要なし。

この面の最後にはボスキャラが登場。このときは鍾乳石がなくなるので，ボスキャラに精神を集中して自機を小刻みに動かしながら攻撃しましょう。このステージは短いうえにクリア時にシールドを満タンにしてくれます。

ボスキャラのひとり（1体？）

ステージ5

指令「大陸横断をし一気に敵陣営へ強制侵入せよ」だって？　なんなんだ，その「一気に」っていうのは。まったく軍の上層部はパイロットを消耗品と思っているんじゃないのか。というわけで舞台は海の上。背景は少し地味だけど，海中から緑の機動兵器が飛び出てくるわ，自分とそっくりの機動兵器が追い越していくわでもう大変。ただ，障害物が少なく画面いっぱいの円運動ができるので慎重にいけばたいしたことはないはず。ステージの最後にはボスキャラの潜水艦が浮上してきてミサイルを撃ってきます。こいつにはナパームの数に余裕のある人はナパームを，ない人は自機を小回りさせながらバルカンを打ち込みましょう。

ステージ6

ゲームも後半に突入。舞台は流れる雲がきれいなはるか上空。空というだけあって障害物がないからいままでどおり大きく回りながら攻撃していればOK，と言いたいところですが，敵のミサイルもいままでのステージとは比べものにならないくらい速い速い。攻略法は画面中心部を中心に小さな円を描くようにしてバルカンを撃ち続けること。なお，ステージクリア時にナパームが全弾補給されるのでナパームをバシバシ使っていきましょう。

ステージ7

前ステージの夕方版といった感じのこのステージ。高速で迫りくる雲にも圧倒されますが，なんといってもすごいのが，カミナリ。遠くでピカッと光ったかと思うと目の前でイカヅチがピシャリと走る。別に自機に対する攻撃ではないし，当たってもダメージにならないけど，心理的になんとなく不安になる。この演出，素晴らしい。攻略法は前ステージとまったく同じです。ただ，今度はナパームが補給されないのでナパームを節約していきましょう。

＊

と，まぁこんな感じでゲームは進んでいきます。残すは3ステージ。果たしてステージ1の敵基地内シミュレーションは役立つのか。敵の最終兵器とはいったい何か。そのへんは自分の目で確かめてください。

最後にこのゲームに対する意見を述べておきますが，このゲームはアクションゲームにもかかわらずスコア表示がないのです。ズームの「ジェノサイド」もそうでしたが，アクションゲームにスコアがないのはどうも目的意識がなくなってしまい，プレイしていて楽しくないのです。初めはエンディングを見たいがために頑張りますが，エンディングを見てしまったら，もうそのゲームを二度とプレイしなくなってしまいがちです。プログラム的には，スコア表示やその計算は非常に簡単なので「ジェノサイド」にしろ「メタルサイト」にしろスコア表示がないのはゲームデザイン上のミスか，勘違いとしか思えないわけです。

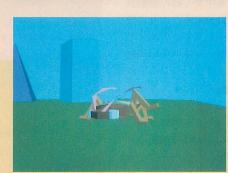
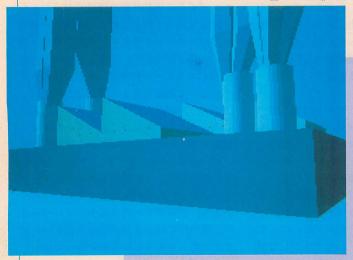
この砂ボコリがけっこうやっかいだ

カミナリの演出がとてもよい

ゲームデザインの話が出てきたついでに述べておきますが，「メタルサイト」はなぜ自機がロボットである必要があるのでしょうか。ゲームアーツの「テグザー」のようにロボット形態と飛行機形態とで機能の差が出るようなゲームデザインをすべきだったと私は思います。このゲームは，オープニングでその機動兵器の機動性なんたらかんたらと言っておきながら，全ステージ通してバックパック背負った飛行形態のロボットを操るだけです。たとえばロボット形態だとスクロールが停止，もしくはゆっくりになって画面にたまった敵を一掃できるとかの工夫がほしかったところです。あと，どうもラストの敵以外は戦っているという感じを得られなかったのが少し残念です。

が，この「メタルサイト」，ゲームバランスは最高ですし，背景のトラップやグラフィックなどの視覚的演出も工夫がなされています。プログラム的にはトップレベルなので次回作ではいま言ったようなゲームデザインのしっかりしたものを作ってほしいものです。

Oh!X Graphic Gallery

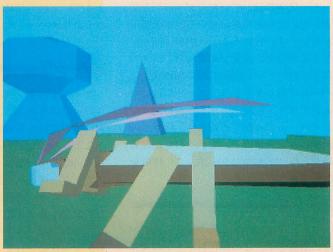
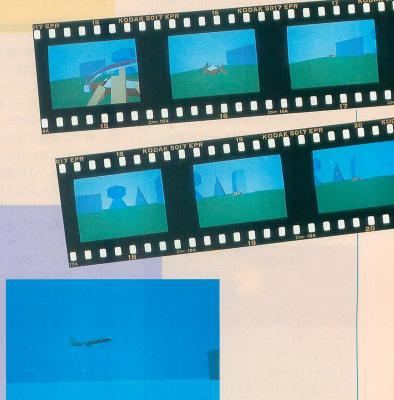
DōGA・CGアニメーション講座

「寺田の教育的指導」

さて，今月登場する作品は，グラフィック研究会のあがた氏の作品「ごきぶりさん」と，北海道の追分高校放送部の「青い空とぼくたちの地球」の2作品からの抜粋（本文92ページ）。

「ごきぶりさん」は，あがた氏のごきぶりに対する愛情にあふれた作品。パステルカラーにあふれた桃源郷のような未来で喜びにあふれた「ごきぶりさん」を見ていただきたい。「青い空とぼくたちの地球」は，NHKの放送コンクール応募用に作られた公害問題をテーマとした実写を含む作品のCG部分。

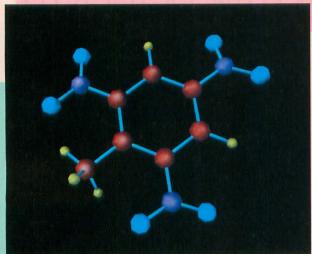
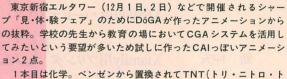
「ごきぶりさん」登場

「ごきぶりさん」のアップ

重々しい工場の煙

工場の上を飛ぶジャンボ機

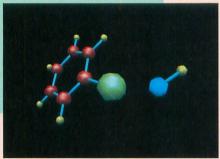
DōGA

「シャープ見・体・験フェア用作品」

東京新宿エルタワー（12月1日，2日）などで開催されるシャープ「見・体・験フェア」のためにDōGAが作ったアニメーションからの抜粋。学校の先生から教育の場においてCGAシステムを活用してみたいという要望が多いため試しに作ったCAIっぽいアニメーション2点。

1本目は化学。ベンゼンから置換されてTNT（トリ・ニトロ・トルエン）が生成されていく過程のアニメーション。2本目は，物理。のどかな草原で大砲を発射し，放物線の性質を勉強するためのアニメーション。このように，目に見えない世界や，実際に行うことが難しいものを映像化・視覚化することで，生徒の理解を容易なものにすることができ，かつ興味を刺激できる。

TNTが完成したところ

45度で発射したときの飛行距離がもっとも長い

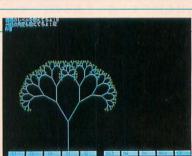
発射角度を変えてみたら

置換が進んでいる過程。2つの分子が接近している

視点の移動の途中。正面から見た放物線と影の動きが等速直線運動であることがわかる

「再帰大作戦」

丹 明彦

C言語で再帰処理とグラフィック機能を使った例。自己相似形から木の枝のような形を構成する。先端の葉は色を変えただけ。

これを3Dに拡張し，レイトレーシングしてみた。まともにやると半月近くかかるので，キャストさんに計算してもらったのが左の写真（1枚30分くらいで終わってしまった）。

再帰処理による木（対称形）

非対称にもできる

C-TRACEで作った木

これは先月と同じ

●新製品レポート

X68000用ハードディスク「Xstorシリーズ」
Xfamily用プリンタ「CZ-8PG1/8PG2/8PK10」

新　仲夫

アイテムからX68000用40MバイトハードディスクXstor40シリーズが発売された。発売されたのは，1台目専用モデルの「HXD040」(118,000円)，2台目専用モデルの「HXD042」(128,000円)の2機種。11月号のペンギン情報コーナーでも紹介しており，購入を考えているユーザーも多いのではないか。使用結果をレポートしたい。

HXD040

●

1台目モデルのHXD040を接続できるのは，ハードディスクを内蔵していないX68000すべて。2台目モデルのHXD042を接続できるのは，ハードディスク内蔵型X68000か，1台目にHXD040を接続したX68000のみ。写真を見てもらえばわかるように，デザインはX68000のマンハッタンシェイプスタイルにマッチ。よって，PROでこのハードディスクを使うのは少し悲しいかもしれない。

接続は最大2台，80Mバイトまで使用可能。将来のことを考えると少ないような気もする。HXD040もHXD042も形状は同じなので2台重ねると結構な高さになる。つまり，マンハッタンビルディングスタイルというべきか。しかし，マニュアルには2段重ねは危険ですと書いてあるので，やらないほうがいいだろう。Xstorを二台つなぐ場合は1台目を本体の横に置くこと。

デザインは

100点満点で85点といったところか。写真ではわからないが，実はX68000本体の奥行きに比べて若干（5cm程度）後ろにはみ出すサイズなのである。うーん，これは横から見ると少し目立つ。もう少し短くしてほしかった。これで－5点というところか。さらに，前面のスイッチがオシャレじゃない。これは，多くの人の感想。それに，スイッチが出っ張っているのも間違って押してしまいそうで不安。さらに－10点。

細かいことだが，X68000本体の下に付いている4つの「足の出っ張り」が上手く収まるようなくぼみが欲しいとの意見もあった。うーん，私は気づかなかったが。しかし，外付けハードディスクにしてはデザインはかなりいいのではないだろうか。

X68000に装着

ハードディスクのスペック

スペックは，表1を見てもらいたい。インタフェイスはSCSIを採用と記してあるが，ご存じのとおりX68000はSCSIではなく拡張SASIのようなものなのでマニュアルを見て混乱しないよーに。平均アクセス速度は23msと通常の40Mバイトタイプのなかでは速いほう（通常は28msが多い）。OSはHuman68kの全バージョンと，マニュアルには書いてないがOS-9でも使用可能だ。フォーマット時に不良セクタの代替処理を行う機能，停電時や切断時のオートパーキングロック機能などは最近のハードディスクには付いている機能で，ないよりはあったほうがいい。

フォーマット～使用感覚

メモリスイッチのセット，フォーマットのやり方は9月号の特集を見ていただきたい。フォーマット時間は，装置の初期化に数分，領域確保に数十分というところだ。

使用した感じは……，はっきり言って他機種と変わらない。劇的に速くなったとの感じもないし，遅いという感じもない。シッピングは，ブレイクキーを押したらかすかに光が点滅。つまり，普通のやつを採用しているらしい。テストの最中ハードディスク上部が少し熱くなってきたのが若干気になった程度だ。

結論として，これは実際に使用してどうのこうのいうよりもデザインがウリだろう。もちろん，基本的な性能はクリアしており，わりと売れるのではないだろうか。

ところで，先月のアイテックのハードディスク紹介記事でも書かれていたが，本当にハードディスクはもう必需品だ。一度使ったらもう元へは戻れない。今後，X68000のハードディスクユーザーはどんどん増えるだろう。そこで，どこかのソフトハウスがX68000用のハードディスクやファイルの管理ユーティリティを出してくれないだろうかと思う今日このごろである。

＜問い合わせ先＞
㈱アイテム　☎03(434)4171

シャープからは，24ドットのインパクト型プリンタ3製品が発売された。特に，24ピンのカラードットインパクトプリンタは純正では初めてである。発売されたのは，カラー80桁のCZ-8PG1(130,000円)，カラー136桁のCZ-8PG2（160,000円)，モノクロ136桁のCZ-8PK10（97,800円）の3製品。X68000，X1，X1turboで使用できる。

●

デザインは，例によってエプソンのOEMのフォルム。純正で初めてカラーのドットインパクト型であるということ以外には，機能的にも特に目新しいものはない。しかし，ドットインパクト型は熱転写型に比べてイニシャルコストは高いが，ランニングコストは低いという特徴があるので，購入を考える人は多いかもしれない。**表2**にスペックを紹介する。

プリンタのスペック

文字フォントはJIS第1，第2，半角，外字，記号，グラフィックなどを搭載。パイカ（文字ピッチが1インチあたり10文字)，エリート(同12文字)，縮小サイズ，スーパースクリプト(上付き)，サブスクリプト(下付き)，影文字，袋文字，縦書き印字などが可能。

印刷可能な用紙は，単票用紙，連続用紙，葉書など。単票用紙と連続用紙の取り扱いはマルチウェイローディング機能で手軽に行える。連続用紙用のトラクタユニットは内蔵されている。また，葉書モードや，高速印字モード，縦書きモードなどの切り替えや，正方向・逆方向の微小送りはすべて操作パネルの操作で行うことができる。7色のカラー印刷は，専用のカラーインクリボンカートリッジを装着して行う。

先にも書いたが，ドットインパクト型はランニングコストが低く，印字速度が速い。また，縦書き，微小送りなど各種の印刷機能を備えているため，コストパフォーマンスは高いと言えるだろう。実際，周りには，すでに欲しがっている人もいるようだ。

<問い合わせ先>
シャープ㈱　☎06(621)1221，03(260)1161

印刷サンプル

CZ-8PG1

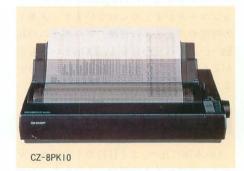
CZ-8PK10

CZ-8PG2

表1　ハードディスクのスペック

	HXD040	HXD042
ユーザー領域（Mバイト）	40	
セクタサイズ（バイト）	256	
トラック密度（TPI）	1500	
記録方式	RLL	
回転速度（rpm）	2358	
平均待ち時間（ms）	12.7	
平均位置決め時間（ms）	23	
データ転送速度（Mバイト/s）	1.5	
SCSI ID番号	1	2
動作時温度/湿度	5～45℃/20～60% RH	
衝撃	5G以下	
寸法（H×W×D)mm	35×136×400	
重量（kg）	2.5	
消費電力（W）	25	
価格（円）	118,000	128,000

表2　プリンタのスペック

（＊1）小数点以下切り捨て

		CZ-8PG1	CZ-8PG2	CZ-8PK10
印字方式/ヘッドピン数		インパクトドットマトリクス/24ピン		
紙送り方式		フリクション・フィード方式，紙幅可変プッシュトラクタフィード方式		
文字種類		パイカ，エリート，縮小，スーパーサブスクリプト 袋文字，影文字，英数字，カタカナ，ひらがな 記号，グラフィック，半角文字，外字 漢字(JIS第1・第2(JIS X0208))		
用紙		単票，連続，葉書		
印字速度（文字/秒）	漢字	53	53	32
	漢字高速	100	100	64
	普通文字	82	82	50
	ドラフト	246	246	150
データバッファ容量（バイト）		256	256	約1K
色数		7	7	1
桁数		80	136	136
寸法（H×W×D）mm＊1		194×496×399	194×609×399	145×593×348
重量（kg）		約 10	約 13	約 8
価格（円）		130,000	160,000	97,800

[特集1]
オペレーティングスタイルの研究

初心者ではよく，パソコンの使い方がわからないという人がいる。しかし，もともとパソコンの使い方というのは自分で考えるものだし，それを考える過程がすなわちパソコンを使うことにもなりうる。与えられたものを使うだけではパソコンの価値はない。使おうとする意志があって初めてオペレーティングという言葉が意味を持つ。それがどういったかたちを取るかはそれほど重要ではない。使いこなそうという姿勢がそれぞれの「オペレーティングスタイル」というかたちで表されるだけなのだ。

OSとオペレーティングスタイル

Nakano Shuichi 中野 修一

OS，つまりオペレーティングシステムという言葉があります。あらゆるコンピュータで基本システムとなる部分がこのように呼ばれています。私たちがパソコンを扱うとき意識するしないにかかわらず，常にOSというもののお世話になっています。具体的には8ビット機のBASICでいえばマシン語モニタまたはBASICそのもの，X68000ならばHuman68kなどがOSにあたります。Humanなどは，性格としてDOSと呼ばれることもあります。

さて，X68000ユーザーにお聞きします。X68000におけるHumanとはどの部分でしょうか。コマンドモードで“A>”のプロンプトを出している部分？　いえ，それはcommand.xですね。OS本体は普段，私たちの目に見えないところで動作しています。メモリ管理，ファイル管理，タスク管理など，面倒なことを引き受けるのがOSの仕事です。config.sysの設定を間違えたときなど，たまに「command.xが起動できませんでした。コマンドを入力してください」などと#マークのプロンプトを出すことがあります。これがHuman68kがもっとも表面に現れる場面といえるでしょう。

プログラミングユーザーに対して，DOSはファンクションコールというものをユーザーに開放し，さまざまなサービスを提供します。Cや福袋のプログラマーズマニュアル，初期X68000のHuman68kユーザーズマニュアルなどにHumanのファンクションコールがまとめられています。メモリの確保やファイルの管理，プログラムの起動……。それらを見ると，DOSというものがどんなものなのかがおぼろげながらわかるでしょう。

ユーザーインタフェイスとは

OS本体に比べ，よく目につくのがユーザーインタフェイスの部分，すなわちcommand.xやvs.xの部分ですね（ユーザーインタフェイスはコンピュータと人間のあいだを取り持つものの意)。これらはUNIXのコマンドインタプリタにならってシェルと呼ばれています。現状ではコマンド（すなわちアプリケーションプログラムなど）を起動するものが広くシェルと呼ばれているようです。

OSの使い勝手はこの部分のデキでほぼ決まってしまいます。特にパソコン上ではユーザーインタフェイスが悪いものは決して高い評価を得ることはできません。むしろユーザーインタフェイスの性能がそのままOSの性能と見られることのほうが多いようです。とんでもないことだと思われる方もいるでしょうが，OSの存在目的のひとつは明らかに人間とのインタフェイスとしての役割ですから，あながち間違っているともいえません。

How to operate “OS”

さて，OSについては「ユーザーはあまり知らなくてもよい」という考え方があります。本来，コンピュータにとって，OSは空気みたいなものですから，ちゃんと動いてさえいれば，「意識してまで呼吸する必要はない」というところでしょうか。

しかし，現状のパソコンでは少しは息を吸う努力をしないと，なにをするにも息苦しくなってしまいます。これはディスクのフォーマットであったり，ファイルのコピーであったり，ハードディスクのインストールであったりといったところでしょう。入門記事を見ても，ちょっとしたことをすると「config.sysを書き換えてください」とか，「～をシステムに組み込みます」などという表現が現れてくるのです。

これがアプリケーションユーザーがOSを使うための動機となります。

一方，パソコンに対して，単なるデータプロセッサとしての意味以上のものを求める場合には，むしろ積極的な呼吸が必要となってきます(MZ，X1やX68000ユーザーはこのような傾向が強いですね)。

いずれにせよ，OSに関する知識は誰にとっても不要ではないわけです。では正しい呼吸法，OSの使い方というものはある

のでしょうか。以下にX68000の場合を例に少々極端なモデルケースを挙げてみましょう。

●A君の場合

学校では，いやおうなくPC-9801を使わされているA君はPDSを駆使してX68000の画面を640×400ドットに調整し，起動時のメモリチェックからメッセージ，はては漢字のフォントまでPC-9801コンパチにしました。コマンド類は当然，同様のものを揃え，CAPSキーはロックしたまま引っこ抜いてあります。これで学校のPC-9801と同じ環境が得られてA君は満足です。あとは1日も早く一太郎コンパチのワープロが現れることを待ち望んでいます。

●B君の場合

B君は“A＞”が嫌いです。バイトに明け暮れ，PC-9801が楽々買える値段でX68000を手に入れたB君は，あくまでもX68000らしさを重視してビジュアルシェルでできる部分は必ずvs.xを使います。工夫を凝らされた彼のビジュアルシェルは強力です。「俺のダブルクリックより速くコマンドタイプできる奴はいない」とうそぶいていました。

ある日「ダブルクリックではファイルの実行と内容確認が両立できないじゃないか」と友人に指摘された彼は，ついに実行型ビジュアルシェル上からタイプ型ビジュアルシェルを起動するという多段式ビジュアルシェルシステムを考え出しました。実行型はダークブルーのスクリーン，タイプ型はアンバーイエローのスクリーンが起動します。いまは高速起動用の増設RAMのためにバイトに明け暮れています。

●C君の場合

X68000はMacintoshのようなマシンだと信じて買ったC君はビジュアルシェルではもの足りないものを感じていました。パソコン通信でMacintoshエミュレータらしきものがあることを知ったC君はどうしても動かしてみたくなり，ついに並行輸入店から安くMacintosh plusを買ってきてしまいました。

MacintoshのROMを吸い上げエミュレータを組み込んでみると結構動くようです。おまけにまっとうなMacintosh plusより速いではありませんか。最近ではSCSIのハードディスクを接続してMacintoshは金食い虫だと嬉しそうに嘆いています。

●D君の場合

大学の研究室でワークステーションを使っているD君は端末不足に業を煮やし，X68000を持ち込んでRS-232Cで接続して端末にしてしまいました。termcap指定が面倒とかROMディスク上のterm.xはなぜかconfig.sysでshellに指定しないとうまくつながらないといいながらも，なんとかうまくつながったようです。

そのうち，HumanのコマンドとUNIXのコマンドがごっちゃになってきてしまいました。そこでD君はHumanのエイリアス機能を使ってHumanのコマンド群をUNIXライクなコマンド名に変えてしまいました。dir/wはls, typeはcat, copyはcpとしたのを始め，process.xはpsにfind.xは不本意ながらgrepとしたようです。もちろんエディタはmicroEMACSです。バッチ処理の貧弱さを嘆き，Bシェルでもいいからシェルがほしいといいながら，いまはC言語（GCC）でUNIXコンパチのコマンドをたくさん作っています。

●E君の場合

E君はその筋の人です。当然，MC68000といえばアセンブラしかありません。もちろんOS-9にも手を出しましたが，IOCSやファンクションコールをうまく処理すればHumanでもマルチタスク化は可能だと信じています。

稲妻のタイピングを誇るE君の前ではヒストリ機能だとかエイリアスなどは無用の長物です。ましてやビジュアルインタフェイスなどは金輪際必要としません。加えて8ビット時代からしみついた性癖のためか，1バイトでも多いフリーエリアのためであれば，どんな環境でも耐えてみせます。最近はエディタやアセンブラ，リンカ，デバッガはROMに焼いてしまおうかと考えています。

＊　＊　＊

さて，これら5人（すべてフィクションです。念のため）は皆それぞれに幸せなパーソナルコンピューティングを楽しんでいるようです。それぞれのオペレーティングスタイルはまったく違います(同じX68000でも……と，いえない部分もありますが)。それぞれが志向するものが違う以上，どれが正しいかなどいえるわけがありません。このほかにも大勢いるであろうと思われる「Load Runner」や「バッチ処理の鬼」，「メニュー小僧」，「ファンクションキーおじさん」，「プルダウンおたく」も同様です。

使い方については「ファイルはこう作成するのが正統派だ」とか「華麗なるディレクトリ構造のあり方」とか「絶対的ユーザーインタフェイス」など，実のところそんなものは存在しようもありません。

OSを使うために，あらゆるマニュアルや解説書を読破したという人は少ないはずです。たいていの場合は，とりあえず触ってみて，わからないところはマニュアルを見て……という過程を繰り返しているにすぎないでしょう。それでも，OSがなんとなく使えるようになっていくというのは，ユーザー1人ひとりが，いわばオペレーティングスタイルとでもいうべきものを持ち始めていることにほかなりません。

もしかしたら，あなたが使っている方法よりもっと便利な機能がちゃんと用意されているかもしれません。しかし，特に残念がる必要もないでしょう。Human68kなどでは実に多彩なユーザーインタフェイスが用意されていますが，そのどれを選ぶかはまったくの自由です。他人から「効率が悪い」とか「キーストロークが多い」とか「直感的でない」といわれても，自分で使いやすければそれに優るものはありません（あくまでもユーザーの立場だからいえることですが）。

以下に続く特集記事も単なるヒントにすぎません。それぞれが，「私のオペレーティングスタイルはこうだ！」と主張している部分もあれば，マニュアルではわかりにくい部分を噛み砕いて説明した部分もあります。まったくの初心者の方はいくらかでも参考に，初心者マークは卒業したという方はそれらをとおして現在の自分のオペレーティングスタイルを見つめ直してみるのもよいのではないでしょうか。

For Beautiful Human68k Life.

基本コマンド攻略法

Nishikawa Zenji
西川 善司
Tajima Hyougo
田嶋 俵吾

Human68kに親しみ，より充実した操作環境を実現するためのもっとも基本的な部分を解説します。まずは基本コマンドの使い方からです。きっと，基本技でもさまざまな応用ができることがわかると思います。

こんにちは，花も恥じらう西川善司です。最近，X68000用に出回っているユーザーメイドのソフトを見ると，そのハードの性能の凄さを再確認するとともに，ユーザーのレベルの高さには驚かされます。しかし，いつでも初心者というものはいるもので，8ビットパソコンからX68000にきた人や，まったくパソコンが初めてという人にとってはOSやDOSとはとらえにくいものかもしれませんね。ここではX68000を買うともれなくついてくるHuman68k，command.xなどの基本的な使い方について述べていくとしましょう。

ただし，コマンド名の後ろにその説明をつけたのではマニュアルと変わりませんから，「やりたいこと→それができるコマンド」の形式で進めていきます。

特定のファイルの一覧を見たい

ご存じのとおり，dirコマンドはファイル一覧を参照するためのコマンドです。作業中，特定の拡張子のついたファイルがディスクの中に存在しているかどうかを確かめたいということがしばしばあります。

BASICでは「.bas」，C言語なら「.c」，バッチファイルは「.bat」というふうに，Humanでは拡張子によって処理を振り分けるものがたくさんありますね。これはOSやアプリケーションが特定の拡張子に対応しているためです。自分でプログラムやファイルを作る場合にも，これにならって系統立った拡張子やファイル名をつけるようにしておくと，使いやすくなります。

たとえば，BASICの音楽プログラムはすべて「m_」で始まり，「.bas」の拡張子を持つとしましょう。音楽プログラムだけを一覧で見たいときにはどうすればいいでしょうか？

こんなときはワイルドカードをdirコマンドに用いてやります。ワイルドカードとは「＊」と「？」の2つの文字のことです。この2文字をファイル名を指定するときに含ませることによって，複数のファイル名，もしくは特定のファイル名を扱うことができるのです。「？」は任意の1文字，「＊」は複数の任意の文字を意味します。

さっきの例はなんとなく「m_＊.bas」でよさそうですね。試してみると，

dir m_＊.bas

でちゃんと一覧が出てきました。

ファイル指定法でもっと細かい例を示しましょう。いま，ディスケットの中に，

abc.bas
abc.x
Abcd.BAS
ABCDE.SYS

という4つのファイルがあったとしましょう。これをAドライブに入れて，

A>

の状態から，

A>dir ＊.bas

を実行してみましょう。すると，

abc.bas
Abcd.BAS

の2つが表示されます（大文字小文字は区別されません)。「＊.bas」の意味はつまり，拡張子が「bas」なら主ファイル名はどんな文字でもいいということです。次に，

A>dir abc?.＊

を実行してみましょう。今度は，

abc.bas
abc.x
Abcd.BAS

の3つが表示されることになります。

今度は拡張子が「＊」で表されていることから，拡張子はなんでもいい！ と指定していることになります。「ABC？」はABCの3文字がついていれば後ろの4文字目はなんでもいい，という意味です。4文字目はなんでもいいわけですから4文字目がなくてもいいわけで，それで「abc.bas」と「abc.x」もターゲットとなったわけです。

「ABCDE.SYS」がターゲットにならなかったのは「ABCD」の後ろに5文字目の

図1 ワイルドカードの例

```
F:\program\xbas>dir /w

 RAM_DISK               F:\program\xbas
    40 ファイル          OK Byte 使用中            OK Byte 使用可能
 ファイル使用量        281K Byte 使用
crc2            bas | m_aria         bas | mac1         BAK | R_mm2
macinto_c       bas | macprint       bas | R_block      bas | R_test3
G_colconv       bas | titleed        bas | M_x68000     bas | title
M_tes           bas | PP             bas | 斌夙          bas | m_GalxF6
CHOICED         BAS | m_gradius      bas | dirtemp          | float3_14
g_palet         bas | pal2           bas | pal3         BAK | pal3
pal3            s   | pal3           o   | pal3         bas | pal3
M_sym9_4        bas | pal4           bas | pal5         bas | G_b12
ti              GSO | pal01          BAK | pal01        bas | situ
scrn$           bas | M_thCross      bas | saizu        bas | opop
F:\program\xbas>dir m_*.bas

 RAM_DISK               F:\program\xbas
     7 ファイル          OK Byte 使用中            OK Byte 使用可能
 ファイル使用量         92K Byte 使用
m_aria          bas      5088  89-06-05  17:31:30
M_x68000        bas       283  89-06-05  17:31:46
M_tes           bas       183  89-06-05  17:31:50
m_GalxF6        bas       284  89-06-05  17:31:54
m_gradius       bas         2  89-06-05  17:32:00
M_sym9_4        bas     65536  89-06-05  17:32:24
M_thCross       bas     17429  89-07-22  15:35:44
F:\program\xbas>
```

「E」があったためです。このことを利用して，４文字以下の拡張子はなんでもいいファイル名の一覧を表示せよ，とするにはどうしたらいいでしょう。答えは，

A>dir　？？？？.＊

です。わかりますか？

ファイルを表示したい

ファイルの中身を見たいときがあります。BASICファイルはX-BASICを起動したあとそのファイルを読んでやって，LISTを実行すれば見ることができます。アセンブラのソースはエディタを，

ed　ファイルネーム

のようにして起動してやれば見ることができます。しかし，もっと手軽にファイルの中身をのぞけないでしょうか。

command.xには「type」コマンドというのがあります。これは，まさにファイルの中身を見るために使います。使い方は，

A>type　ファイルネーム

で先ほど説明したワイルドカードも使えます。ただ大きいファイルになると当たり前ですが，1画面内に収まらず，スクロールアウトしていってしまいます。[CTRL]+[S]で表示の一時停止が可能ですがどうも不便です。そういうときには，「more」を使います。使い方は，

A>more　<　ファイルネーム

です。これを実行すると表示が画面いっぱいになるとキー入力待ちとなり，表示を一時停止してくれます。dirコマンドの「/P」と似ているといったらわかるでしょうか。「<」は「more」コマンドがフィルタ系のコマンドのためです（<は省略可)。<は←の意味で，キーボードの代わりに後ろに書かれたファイルから文字を与えることを表しています。

さて，いままでは文章のようなファイル，テキストファイルを表示するためのコマンドで，ADPCMのデータファイルや，拡張子が「R」や「X」の実行形式のファイルを表示できなくはないのですが，どうも画面が乱れたりして気持ち悪いですよね。そんなときにはこれ，「dump」です。

A>dump　ファイルネーム

のようにして使います。８ビットパソコンなどでマシン語モニタに入って，

＊D　？？？？　？？？？

のようにしてメモリ内の表示をしたことがあるでしょうか？　この「dump」はそれとよく似ていてファイルの内容を16進数と，それに対応する文字で表示します。このコマンドも表示が１画面に納まらないときはどんどんスクロールアップしていってしまいます。うーん，どうにかならないでしょうかね。先ほど「more」というのが出てきましたね。これはフィルタ系のコマンドといいましたが，フィルタっていったいなんなんでしょう。

あるコマンドを実行して，その実行結果を直接出力（多くの場合は画面に表示）せず，少し手を加えてから出力させる機能があり，この「手を加える」のがフィルタ系コマンドの役目です（フィルタ系のコマンドは「find」，「more」，「pr」，「sort」の４つが用意されています)。

「more」は１画面ごとに区切って表示をする，という役目のフィルタコマンドだったんです。そのほか，意外かもしれませんが，Oh!Xでリスト打ち出しに多用されている行番号は実は，

find/v　文字列　ファイル名　>prn

のようにフィルタコマンドfindを使って出力されています（文字列の部分にはファイル中に絶対存在しないものを指定する)。

フィルタは“/”と共によく使われます。これは“/”の前のコマンドの表示を横取りし，後ろのコマンドに渡す働きをします。

A>more　<　ファイルネーム

は，

A>type　ファイルネーム　|more

と同じことなのです。もう，わかりましたね。「dump」コマンドも同じように実行してやればいいのです。

A>dump　ファイルネーム　|more

です。「/p」があるのでほとんど意味がありませんが，dirコマンドを，

A>dir　|more

として使うことも可能です。

あ，ひとつ重要なことをいい忘れてました。フィルタコマンドはその実行に作業用の「テンポラリファイル」というものを自動的に作成しますので，ディスケットのプロテクトノッチにシールが貼ってあったり，ディスクの容量が足りなかったりするとエラーが発生します。また，このテンポラリファイルはコマンド終了時に自動的に削除されます。

階層ディレクトリってなんのこと

５インチ2HDのディスクは容量が1Mバイト以上もあるので，１枚のディスクにかなりの数のファイルを入れておくことが可能です。また，ハードディスクともなるとその何十倍ものファイルを入れることができます。なにも考えずたくさんのファイルを入れてしまうとdirコマンドを実行したりしたとき，目的のファイルを画面から読み取る前にスクロールアウトしてしまったりして大変不便です。まぁ，

A>dir　/w

(画面いっぱいにファイル一覧を表示)や，

A>dir　/p

（ファイル一覧が画面いっぱいになるとキー入力を持って続きを表示する）でなんとかならないこともありませんが，なんとか系統立ててファイル管理をしたいですね。そこで階層ディレクトリです。たとえば，

のようにファイルをディレクトリ単位で管理することを階層構造と呼びます。

「ルートディレクトリ」は略して「ルート」などと呼ばれますが，これは階層ディレクトリ構造の大本です。ルートは木でいうと根本の部分みたいなもので，ほかのディレクトリ（サブディレクトリ）は枝に当たり，ファイルは葉とでも考えてください。ちなみに、ディレクトリ構造をこの目で見てみたい，という人のためにtreeというコマンドがあります。使い方は，

A>tree

です(Human68k ver.1.∅にはありません)。

すべてのファイル操作は基本的にはいま，自分のいるディレクトリ内で行われます。ですから，

A>type　＊.bas

としても別のディレクトリにあるファイルを見ることはできません。

「いま，自分のいる」っていったって，どうやったらそれがわかるんだ？　と思った人も多いはずです。では，適当なシステムディスクを立ち上げて，

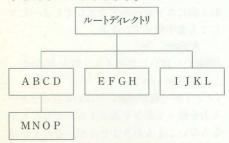
A>cd

を実行してみましょう。ルートにいる人は，

A：¥

と表示されたはず。そのほかのサブディレクトリにいた人は，

A：¥　ディレクトリ名

と表示されたでしょう。これで，いま，自分が操作できるディレクトリがわかったわけです（この「いま自分のいるディレクトリ」をカレントディレクトリと呼びます）。それでは，別のディレクトリに移るにはどうしたらいいのでしょう。

という，ディレクトリ構造をしているディスクを想定しましょう。いま自分がEFGHにいたとしてルートへ移るにはどうしたらいいでしょう。それは，

A>cd ¥

としてやればいいのです（ちょっとわかりにくいかな，でもあと，いくつかの例を見ていけばわかってしまうと思うよ）。次にルートからIJKLへ移るにはどうしたらいいでしょう。

A>cd IJKL

です。この状態からMNOPへ移るにはどうしたらいいでしょう。

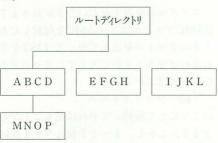
A>cd ¥ABCD¥MNOP

が答えです。では，少し詳しく見ていくことにしましょう。

ひとつ下層のディレクトリに行く場合は，cdの後ろにディレクトリ名を書いてやるだけです（先ほどの例の，ルートからIJKLへ移ったときの場合）。これは比較的わかりやすいと思います。

さて，少し難しそうなのが最初と3番目の例のやつ。これはディレクトリの絶対指定といわれ，あたかもルートから下層ディレクトリへ順番に降りていくように指定してやる方法なのです。ですから，

A>cd ¥ABCD¥MNOP

は，

のように上から順番に指定してMNOPまでたどり着いたと考えても結構です。最初

環境ってなんだ?

田嶋 俵吾

雑誌なんかでOS関係の記事を見ていると，「環境を整備する」という言葉が目についたりする。普通「環境を整備する」といえば，日曜日にドブのゴミ掃除をやったり，心ないドライバーが投げ捨てていった空き缶を回収したりするような作業を思い出すもんだ。それがなんでコンピュータと関係あるんだろ。あ，そうかそうか。机を片づけてマウスを動かせるスペースを空けたり，部屋中に散乱しているディスケットをケースに戻したりするのかな？　いわゆる「68のための奉仕」というヤツ。などとオチャラケていてはイケナイ。

一般の「環境の整備」というやつが，良きにつけ悪しきにつけ，住みよい街を作ろうとしているのと同じように，コンピュータでいう「環境の整備」も，より快適にOSを使えるようにするためにあるのだ。

さて，Human68kやMS-DOSを見た場合，環境整備は大きく2つに分けて考えることができる。ひとつはFM音源を使えるようにしたり，ASK68Kを使えるようにすること。つまり，自分がマシンを使っていくのに必要なものを用意することだ。これはconfig. sysというファイルで「使うゾ！」と宣言しておけば，自動的に用意されるもので，ここでは触れない。もうひとつが本命中の本命で，これから説明する環境変数というやつだ。

誰のための環境変数?

環境変数という言葉は，「環境」+「変数」と考えることができる。これまでのノリで読むと，「環境を整えるための変数」ということになるだろう。そいじゃ，いったい誰のための環境をどうする変数なのか。ここが問題だな。

実はこの環境変数は，command. x のための環境を整える変数なのだ（いまはそう思っておいてかまわない）。ではおもむろにX68000に付属してきたHuman68k のディスクを起動してみることにしよう。

おっとvs.x（ビジュアルシェル）が起動してしまったゾ。そうか，そうか，そうだった。ここでは環境変数がなんなのか説明しづらいから，command. x を起動してみてもらいたい。戦車の絵にマウスカーソルを合わせてダブルクリックすればいい。画面が真っ暗になって，

A>

などという，なんの愛想もない文字が現れたことと思う。vs.xのビジュアルな画面とはまったく違う。ここではアイコンをダブルクリックする代わりに，命令を直接打ち込んで使うことになっている。どんなファイルが入っているのかな？　と思ったら，

dir

と打ち込むというようにだ。

さて，「dir」と打ち込んだら，画面にズラズラと文字があふれてきたことと思う。それぞれの行に表示されているのが，ビジュアルシェルでのひとつのアイコンに相当する。あのビューティフルでユニークなアイコンが，こんな情けない姿になってしまったわけだ。

表示された文字のなかに，

BIN　<DIR>　……

ってのがあるのがわかるだろうか。これはBINってのはディレクトリ（Directory）で，その中にまだファイルが入ってるよという印だ。中にまだあるってんだからのぞいてみよう。これは，

dir bin

とやればいい。画面に収まりきらないほど文字が表示されたことと思う。

〜　x ……

と表示されているのは，実行できるファイルだ。試しに実行してみよう。

ed

と入力してみてもらいたい。エディタedが起動しただろう。終了するにはESCキーを押してからQを押せばいい。

command. xってやつはよくできているようで間抜けなところもある。0ドライブのディスクを抜いて1ドライブに入れ，ワープロのディスクを0ドライブに入れてみよう。ここでもう一度，

ed

と入力してみてもらいたい。ほら，

コマンドまたはファイル名が違います

と出た。edがどこにあるのかcommand. x に

の例のルートへの戻り方も同じように考えることができます。

A>cd ¥

はいちばん上のルートから下へ降りていかなかったと考えるわけです。

絶対指定があるなら相対指定もあるのです。ひとつ上層のディレクトリに戻るときに，

A>cd ..

としてやるのがその相対指定です。ですからいままでの例でいくと，ABCDにいるとき「cd ..」を実行してやればルートへ戻れます。

このディレクトリの相対/絶対指定は「cd」コマンド以外にも適用されます。たとえばディレクトリABCDのファイルの内容をすべて見たいときは，

A>type ¥ABCD¥*.*

または，

A>type ABCD

としてやります。そのほか，

A>type ..¥test.s

とか，Human68kのファイル指定はMS-DOSより柔軟なため，恐ろしいことに，

A>type b:

とBドライブの内容をすべて表示するなどの指定もできます。

そうそう，カレントディレクトリを確かめるには「cd」を使う以外にプロンプトを変更してやることでも可能です。

A>prompt pg

としてみてください。詳しく知りたい人はマニュアルの「prompt」命令の項を見てみましょう。工夫しだいでいろんなことができますよ。

外部コマンドと内部コマンド

付属の「Human68kユーザーズマニュアル」をパラパラとめくってみましょう。コマンドの解説欄の右上に「内部コマンド」とか「外部コマンド」とか書いてあるでしょう。これってどういう意味なんでしょうね。答えは簡単。内部コマンドとはcommand.xがその内部に持っている(いつでも使える状態にある)コマンドです。それに対して外部コマンドはその名のとおり，そのプログラムが外部に存在するコマンドのことです。

ですから外部コマンドを実行する際にはその外部コマンドのプログラムの入ったディスケットが本体に挿入されていなければいけません。本文中の議論の中心となった「type」と「more」ですが「type」は内部コマンドなのに対し，「more」は外部コマンドです。また，内部コマンドを増やすことはほとんど不可能ですが，外部コマンドはいわばひとつのプログラムですから，プログラムの作れる人ならばいくらでも増やせます。

はわからないので，実行できませんと悲鳴を上げているのだ（にしては偉そうなメッセージだけどね）。

実はcommand.xが実行できるのは，「dir」と入力したときに表示されるものの中にある「～ x ……」だけなのである。「～ x ……」と書くのは面倒なので，以後これを「～.x」と書くことにする。いま dir とやっても ed は見つからない。だから実行できないのだ。

さっきは実行できたじゃないか！ Humanのディスクを0ドライブに入れてたときにdirとやってもed.xは表示されなかったぞ。確かBINの中にあったはずだ。ごもっとも。実はそれこそが，環境変数の妙なのであり，環境変数が大切なわけもここにある。

忘れちゃいけないpath変数

command.xがdirとやったときに表示される中にある「～.x」しか実行しないのは先に書いたとおり。この「dir」とやったときに表示されるディレクトリのことを「カレントディレクトリ」という。つまりcommand.xはカレントディレクトリにあるファイルしか実行できないわけだ。

しかし，それじゃああんまり不便じゃないか，と考えた輩がいたんだな，きっと。dirで表示される中に指定されたファイルがない場合は，○○ディレクトリと，△△ディレクトリと，……ディレクトリもついでに探してくれるよう，設定しておけるようになっているわけだ。さっき，「ed」とやって BIN ディレクトリの中にあるed.xが実行できたのも，ない場合はBINディレクトリも探しなさいと設定してあったからにほかならない。

ではどこに設定してあるのか。そいつを調べるには，

set

または，

path

とキーボードから打ち込んでみればいい。

path=A:¥;A:¥SHELL;A:¥SYS;…

などと表示されたことと思う（これはHuman68k Ver.2.0の場合。ほかのバージョンでは表示が若干違うかもしれない）。path（パス）ってのは小径という意味だ。つまり，入力されたファイル名がカレントディレクトリにない場合，command.x はこの小径をたどって「ここにはあるかな？ そいじゃこっちにはあるかな？」と探していくわけだ。

探すべきディレクトリは，

path=ディレクトリ;ディレクトリ……

と，「path=」に続けて';'で区切りながら列挙されている。先の例でいうと，カレントディレクトリになければ次にAドライブのルートディレクトリを，そこにもなければAドライブのSHELLディレクトリを，そこにもなければ……を探しなさいと指示してあることになる。

ここでAドライブというのは0ドライブのこと。Human68kではA，B，C……という英字でドライブを区別することになっている。

さて，表示された「path=……」の中に，AドライブのBINディレクトリを探しなさいという指示を見つけることができるだろう。これのおかげで，ed.xがAドライブの BIN ディレクトリにあるときは実行できたわけである。当然，

path=……;B:¥BIN;……

なんてのはないから，ed.xの入っているディスクを1ドライブ（Bドライブ）に入れたときには実行できなかったわけだ。

Human68kには実に多くのファイルがついてくるが，それをすべて必要とする人はあんまりいないだろう。フロッピーディスクには1Mバイトしか入れられないから，誰かが作ったちょっとおいしいプログラムなんかも入れておこうと考え始めると，たちまちいっぱいになってしまう。

そこで，最初についてきたコマンドの中から必要なものを選んでDOSディレクトリに整

図1 環境変数の例

```
temp=f:
comspec = a:¥shell¥command.x
lib=a:¥c¥lib
include=a:¥c¥include
windex=a:¥windex
path=a:¥;a:¥bin;a:¥basic;a:¥c;a:¥wp;a:¥c;a:¥sound;a:¥music;a:¥sys;a:¥shell;a:¥et
c;a¥quickstart;a:¥dic;a:¥windex;f:¥
prompt=$E[s$E[1;H$E[45m$E[0K$E =====  $V  ====  < $P >$E[80C$E[16D$D$E[u$E[A$E[B
$N$G
```

隠れファイルってなんでしょ

市販ソフトのなかにはHuman68kから立ち上がるのに，そのディスクに「dir」しても，「ファイルがありません」なんて出てくるときがありますね。え？　市販ソフトだからできる芸当なんでしょ？　だって？　とんでもありません。普通のユーザーの私たちにだってできるんですよ，外部コマンド「attrib.x」を使って。

Human68kには書き換え可能なファイル属性が2つあるのです。それは，

H……ファイル名を隠す属性

R……読み出し専用，削除を不可能にする属性

の2つです。「attrib」（アトリビュート）コマンドは，

A>attrib　属性スイッチ　ファイル名

のようにして使います。属性スイッチとはさっき挙げた2つの属性をかけるのか，それとも解除するのかを指定してやるスイッチで“＋”で設定“－”で解除です。このあとに先ほどの属性の1文字をくっつけてやると属性スイッチの完成です。command.xを不可視にしたいならば，

A>attrib ＋h command.x

とします。解除は，

A>attrib －h command.x

です。不可視や読み込み専用にするとファイル削除の命令などにも影響を受けなくなりますのできわめて重要なファイルはこれらの属性を設定してやるとよいでしょう。不可視にするとdirでファイル名は出てこなくなりますので注意が必要です。

また，この「attrib」には便利な機能として，ディスケットの中のファイル一覧を属性付きで表示することができます。このときばかりは不可視の属性のファイルを見ることができます。ですから市販ソフトなんかをBドライブに入れて，

A>attrib　B：＊.＊

を実行してみると思わぬ隠れファイルが見つかるかもしれません。

子プロセスと親プロセス

Human68kではあるプログラムを実行中に（親プロセス）一時的に別のプログラムへ処理を移す（子プロセス）ことが可能なのです。たとえばマスターのシステムディスクを立ち上げるとvs.xが起動しますよね。ここで戦車のアイコンをクリックして，command.xを起動してください。この時点で，vs.xが親プロセス，command.xは子プロセスです。さて，

A>basic

理し，友人からもらったりダウンロードしたPDSのコマンドをTOOLSディレクトリに整理したりすることになる。こうして，必要なコマンドだけが入った自分専用のシステムディスクを作るわけだ。ほんとにフロッピーディスクだけで扱おうとすると苦労する。ハードディスクが売れるわけもわかるね。

さて，上のようにファイルを整理し，図1のようなディレクトリを作ったとしよう。これでシステムディスクが完成したと安心してはいけない。大切なことを忘れている。環境変数pathの設定だ。図1では，もともとBINディレクトリに入っていた，Human68kについてきたコマンドをDOSディレクトリに整理し直している。さらに，自分専用のツールとして，いくつかのコマンドをTOOLSディレクトリに整理した。そこで，これらのディレクトリ内にあるファイルを実行できるよう，pathを設定しなければならない。

図2　自分専用のディレクトリ構成

```
¥
|-CONFIG.SYS
|-AUTOEXEC.BAT
|-USKCG.SYS
|-KEY.SYS
|-COMMAND.X
|- ……
|
|-CONFIG
|    |-BEEP.SYS
|
|-SYS
|    |-ASK68K.SYS
|    |-RAMDISK.SYS
|    |- ……
|
|-DOS
|    |-COPYALL.X
|    |-FORMAT.X
|    |- ……
|
|-TOOLS
|    |-WIDTH.X
|    |-DRV.X
|    |- ……
|
|-ASK
     |-ENV1.ASK
```

環境変数設定の方法

環境変数を設定するには，環境変数を表示するときに使った「set」コマンドを使用する。これはcommand.xが内蔵しているコマンドなので，どこを探してもset.xなどというファイルは見つからないはずだ。

setコマンドは，

set　環境変数名＝その内容

という書式で使用する。図1のディレクトリ構成でpathを設定するなら，

set path＝A:¥;A:¥DOS;A:¥TOOLS…

などとなる。要は，SETに続けて，環境変数が表示されるときの書式そのままに書けばいい。pathの場合は専用のコマンドもあるのだが（その名もpath），環境変数の一環としてとらえるとsetコマンドが使えるわけだ。ただし，

set PATH＝A:¥;A:¥DOS;A:¥TOOLS…

や，

set path ＝A:¥;A:¥DOS;A:¥TOOLS…

のように大文字を使ったり空白を加えると違う環境変数になってしまうので注意してほしい。Humanのpathは小文字のpathだ。

逆に，なんらかの理由で環境変数をリセットしたい場合には，

set　path＝

とする。これは，環境変数pathに「なにもなし」を設定する，と読めるだろう。このとおり実行したあと，環境変数を表示してみてほしい。pathという環境変数がなくなってしまっているはずだ。

プログラムの中には，コンフィギュレーション用のファイルなど，自分の実行に必要なファイルをpathで指定されているディレクトリから検索するものがある。これらを別のディレクトリに保存した場合はpath変数にそのディレクトリをつけ加えておくのを忘れないようにしたい。

環境変数の役割

先にとりあえずということで，環境変数はcommand.xのための環境を整える変数だと説明した。ここで改めて環境変数の本当の役割を説明しておこう。

例として次のようなファイル「～.x」を考えてみよう。名前がないのは不便なので，仮にcc.xという名前にしておく。cc.xは単体では実行できず，数十個のファイルを参照しなければならないものとしよう。しかもこれら数十個のファイルは，cc.xを使う人がどんな名前のディレクトリに格納するか，まったくわからないものとする。

さて，cc.xは参照すべき数十個のファイルがどのディレクトリにあるか，どうやって知ればいいのだろう。ひとつの手としては，環境変数pathにこれら数十個のファイルが入っているディレクトリを書いておくという方法

として，BASICを起動してください。立ち上がりましたか？　この時点でBASICは子プロセスで，command.xが親プロセスです。また，command.xにとってvs.xはやはり親プロセスです。

BASICで作業しているうちにディスクが足りなくなりました。どうしますか？　こんなときにこそ子プロセスです。BASICから，

　!

と打ち込んでリターンしてください。command.xが立ち上がりましたね。さあ，formatでもなんでも立ち上げてください。もうおわかりかと思いますが，このcommand.xはBASICの子プロセスになっています。

この状態から，

　A>exit

としてください。あら不思議。BASICに戻っちゃった。メモリを増設していれば，B

図2 プロセス表示例

がある。しかし，数十個のファイルを参照するのに，「BINにはない，SHELLにもない，TOOLSにもない，……，あ〜やっと見つかった！」などといちいちやっちゃいられない。待ってるほうはイライラしっ放しだ。

それじゃどうする。そう，環境変数を使えばいいのである。これら数十個のファイルが入っているディレクトリを，

　set include=b:¥c¥include

ってなぐあいに設定しておいたなら，cc.xは「includeに設定されているディレクトリはなぁにかな？　ほう！　b:¥c¥includeか。じゃあ俺が参照すべき数十個のファイルはここにあるんだな」といとも簡単に調べることができるわけだ。これで誰がどこに数十個のファイルをしまっておこうが大丈夫だ。cc.xはちゃんとそれを見つけ出すことができる。

このようにファイルの中には，自分が必要とする情報を環境変数の形式で設定しておいてくれと要求するものがある。環境変数は，あなたとHuman68kとプログラムの3つを結ぶ掛け橋だといえるだろう。そうそう，cc.xってのは，あのぶ厚いマニュアルで意表を突くコピープロテクトを実現した，シャープ純正のX68000用Cコンパイラのことである。

ちょっとおいしい環境変数の利用法

Human68k Ver.2.0にはヒストリドライバhistory.xが付属している。ヒストリというのは，ユーザーが打ち込んだコマンドを覚えておいてくれるものだ。ヒストリ関連の操作は非常に多彩で，コマンドに与えたパラメータのヒストリを利用できたり，「A:¥BIN」で始まるヒストリだけを表示するなんてこともできる。

このような機能が盛りだくさんのhistory.xは，環境変数に対しても操作しやすいような配慮がなされている。その最たるものは環境変数をコマンドラインで展開できる機能だ。command.xで，

　set path=%path%

と打ち込み（まだリターンキーは押さない），ESCキーを押してから“/”を押してみよう。ほら，コマンドラインが，

　set path=A:¥;A:¥SHELL;……

に変わっただろう。環境変数pathに設定されている内容で，%path%が置き換えられたのだ。環境変数pathに新しいディレクトリを追加したければ，ここでカーソルキーを使って新しいディレクトリを追加したいところまで移動し，そこで打ち込めばいい。

pathからディレクトリをどれか削除したい場合も同様だ。変更したらリターンキーを押して，pathの更新は終了となる。

もし，pathの先頭か最後に新しくディレクトリをつけ加えたいなら，操作はもっと簡単だ。先頭にMYDIRというディレクトリを加えるなら，

　set path=A:¥MYDIR;%path%

でOKだし，最後につけ加えるなら，

　set path=%path%;A:¥MYDIR

となる。いずれの場合も，リターンキーを押したと同時に内部で%path%が展開されるので，現在のpathの内容につけ加えることができるようになる。

リターンキーを押したと同時に%でくくった環境変数が展開されることを利用して，次のようなことも可能だ。

　dir %include%

includeは先に出てきたように，cc.xが実行時に必要とする数十個のファイルを収めたディレクトリをセットしておく環境変数である。上の命令は，その数十個のファイルが収めてあるディレクトリの中を表示するという意味だ。さらにファイルの中身を画面に表示するtypeコマンドを使って，

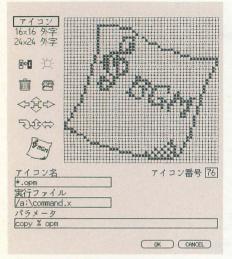
　type %include%¥stdio.h

なんてこともできてしまう。げに恐ろしきはhistory.xかな。

history.xを使うことにすると，使用できるメモリが8Kバイト減ることになるが，たかが8Kバイトで便利な機能を満喫できるわけだから，これを使わない手はないだろう。

以上Human68kでの環境変数の意味とその利用法について考えてきた。環境変数というものがわかってもらえただろうか。環境変数は2つに分けることができる。ひとつはpathのように設定されていないとどうしようもないもの，もうひとつはある特定のプログラムを使う際に必要となるものである。

pathのようにHuman68kの(command.xの)動作に直接関係あるものは，Human68kが起動するときに同時に設定されるのが望ましい。また，あるプログラムを利用するのに必要な環境変数群をいちいちキーボードから入力するのも面倒な話だ。これらの問題を解決するのがバッチファイルである。環境変数がどういうものかわかったら，次はバッチファイルでその効果的な利用方法を会得してほしい。

図3 ファイル名は%の部分に入る

ASICからエディタを呼んだり，コンパイルしたりといったことが楽々できてしまうのです。

ではBASICから，

exit( )

または

system

と打ち込んでリターンしてください。あらあら，command.x に戻っちゃいましたね。では再びこの状態から，

A＞exit

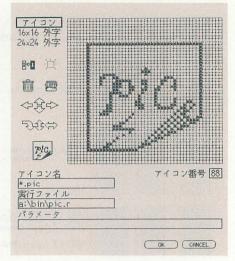
を実行してください。おっと，今度はvs.xビジュアルシェルに帰ってきてしまいました。

このように，ちょっと，command.x でしたいことがある，など，「ちょっと」したプログラムの行ったり来たりに，子プロセスと親プロセスの概念はとても役に立ちます。親プロセスに戻るときにはなんといってもあのうっとうしいディスクアクセスがありませんからね。

あ，ちなみに，根っこのプロセスからexit としてもエラーが出るだけです。念のため。

VS.Xが簡易メニューに早変わり!?

vs.xというのは立ち上がりが遅いので案外嫌われがちです。でも使い方によってはなかなか面白いことができます。たとえばOPMファイルを始めとした音楽プログラムだけを集めた音楽ディスクを作るとします。OPMファイルを演奏するには，OPMドライバが組み込んである状態で，

図4 ファイル名はそのまま渡される

A＞copy　ファイル名　opm

としなくてはなりませんね。まあ，

copy %1.opm opm

というようなバッチファイル，「play.bat」みたいのを作っておけば，

A＞play　OPMファイル

で曲が聞けなくもありませんが，ファイルネームを打ち込んだりするのはまだ少し面倒くさいですよね。そこで，vs.xのあのマウスオペレーションのメニュー機能を生かせないでしょうか。では，ここからはOPMファイルをvs.xのメニュー機能を生かして演奏する，ということにターゲットを絞って話を進めます。

vs.xの画面右真ん中下あたりに机の上に本やら三角定規が置いてあるアイコンがありますね。これをマウスカーソルで指して，右ボタンを押しながら「Icon Maintenance」を選んでください。音符の絵でもメイン画面に描いたら，「アイコン名」のところをクリックして，

*.opm

とキータイプしてリターンを押します。次に「実行ファイル」をクリックして，

/command.x

と入力してください。最後にパラメータをクリックして，

copy % opm

と入力してください。それでは「OK」をクリックして「Icon Maintenance」を抜けます。これで，音楽vs.xが完成です。

そのディスクにOPMファイルをたくさんコピーしてやったあと，OPMファイルのアイコンをクリックしてみましょう。ほら一度COMMAND.Xが立ち上がったあと，音楽が鳴り始めましたね。その後すぐに VS.X に帰ってきました。音楽はまだ鳴ったままのはずです。さて，それでは，いままでの作業はいったいどんな意味があったのか順番に見ていくことにしましょう。

「Icon Maintenance」の「アイコン名」は，まさしく，いまエディットしているアイコンがどのファイルに対応するかを示すものです。「*.opm」はワイルドカードによるファイル指定で拡張子が「opm」のファイルすべてを対象とするという意味になります。

「実行ファイル」はそのアイコンをクリックしたときにどのファイルを実行するかを指定するものです。ここでは「/command.x」としていますので，ダブルクリックでcommand.xがひとまず起動します。「/」は実行ファイルの実行終了後すぐにvs.xへ戻ってくるようにするための隠しコマンドのようなものです（「/」を指定しないと実行終了後に「vs.x : press mouse button or key」と出てしまい，少し情けない）。

「パラメータ」とは，「実行ファイル」につけるパラメータのことです。ここでは「実行ファイルに「command.x」で，「パラメータ」に「copy % opm」としていますから，コマンドラインから，

A＞command/c copy % opm

とやったのと同じになります。これは，つまり，子プロセスでcommand.x を立ち上げて，その直後に「copy % opm」を実行することと同じです。「%」が「%1」のようなパラメータナンバーを指定していませんがアイコンでクリックしたファイルネームを参照するにはこうするものなのだ，と覚えておいてください。

電脳倶楽部などで有名なグラフィック圧縮展開プログラム「pic.r」を用いてvs.xをグラフィックギャラリーにしてやることもできます。この場合は起動ファイルを「pic.r」にするだけで，パラメータは必要ありません。これで，アイコンをクリックすると絵が出るようになります。

＊　＊　＊

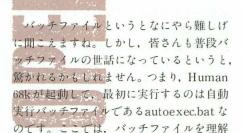
これであなたはもう“Human68k PRO-68K”です。Human68kを使いこなせるようになったら，せっかくの68000MPUです。今度はマシン語をマスターしてプログラムが組めるようにがんばりましょう。

コマンドひとつで全自動

基礎から学ぶバッチファイル

Izumi Daisuke
泉　大介

MS-DOSの入門書では高等テクニックとして扱われるバッチ処理もわかってしまえば簡単なものです。ここではバッチ処理の基本を実例を挙げて解説します。そのほか，バッチファイルで使用すると便利なコマンドなども作ってみましょう。

バッチファイルというとなにやら難しげに聞こえますね。しかし，皆さんも普段バッチファイルの世話になっているというと，驚かれるかもしれません。つまり，Human68kが起動して，最初に実行するのは自動実行バッチファイルであるautoexec.batなのです。ここでは，バッチファイルを理解し，自分のものにしてみましょう。

バッチファイルってなあに

辞書をRAMディスクに転送する，アプリケーションを使うために必要な環境変数を設定する，など，Human68kを利用していくうえでは，決まりきった手順をキーボードから入力しなければいけません。が，毎回同じ手順をキーボードから入力するのは，キーボードに不慣れな人はもちろん，ブラインドタイプでガシガシとキー入力できる人でさえ面倒な作業です。しかし，このような一連の操作も，ひとつのコマンドで実行できるのであれば便利でしょう。それを可能にしてくれるのがバッチファイルです。

バッチ（batch）とは，「一括処理」という意味です。辞書をRAMディスクに転送し，ワープロを起動して，ワープロが終了すれば辞書を元のディスクに戻す，といった一連の処理にeditなどという名前をつけておき，editと入力するだけでこれら一連の作業を一括してやってしまおうというものです。ここではHuman68kにおけるバッチ処理の使い方について勉強しましょう。

バッチファイルの作り方と使い方

バッチファイルは，基本的にはcommand.xで入力する命令をそのまま列挙したものです。試しに，Aドライブのディレクトリ BINのファイルリストを表示するバッチファイルを作ってみましょう。バッチファイルはコマンド，すなわち文字の集まりですから，ed.xを使うといいでしょう。では，cmds.batという名前のバッチファイルを作りましょう。バッチファイルの拡張子はbatです。以下のようにcommand.xの命令を書けばバッチファイルができます。

```
dir a:¥bin /w
```

ESCキーを押してEを押せばセーブされます。では，cmds.batを実行してみましょう。Human68kのコマンドラインから，

```
cmds
```

と入力すればOKです。AドライブのBINディレクトリのファイル名が表示されましたね。このようにバッチファイルは.xファイルと同じように実行できます。アセンブラやCコンパイラを使う必要もなく，エディタでコマンド名を列挙していくだけで作成できるという簡易さもバッチファイルの魅力のひとつです。

バッチで使える制御命令

単にある命令に別の名前を与えるだけなら，バッチファイルが使えてもなにも面白くはありません。バッチファイルが「おいしい」のは，その中でいろいろなバッチ用命令が使えることです。これはBASICの命令を簡略化したようなもので，数も多くありませんが，使い方によっては便利に使える命令です。多くはコマンドラインでも使用可能です。

●for %変数 in（項目セット）doコマンド

これは主に，同じ作業を複数のファイルに対して行うときに使います。以下にカレントディレクトリにある.basという拡張子が付いたファイルを順次表示（type）するバッチ命令を紹介します。

```
for %a in (*.bas) do type %a
```

forはこのように使います。inの後ろにカッコでくくってあるのが対象とするファイル群です。対象とするファイル群にマッチするファイル名がひとつずつ順番に%aにセットされます。そうして，そのファイルに対しどのようなコマンドを行うのかをdo以降で指示します。%の後ろは別にaである必要はありません。任意の1文字が使えます。要するに「%文字」は変数のように使われるわけですね。

これは，ファイルから文字列を検索するコマンドfindなどを使う場合重宝します。たとえばC言語などで，ある変数名が定義してあるのはどのファイルだったかな？　と確かめる場合などです。注意したいのは，カレントディレクトリ以外のディレクトリにあるファイルをカッコ内で指定した場合です。このとき%変数にセットされるファイル名にはパス名が含まれていませんので，

```
for %a in (b:*.c) do type b:%a
```

というように，do以降でもパス名を書いておく必要があります。正確には，forはカッコの中に列挙された文字列にマッチするものを%変数にセットして，do以降を実行します。つまり，カッコ内にはファイル名以外も書けるということなので，

```
for %a in(type dump)do %a test
```

とすれば，testというファイルの内容をまずtypeコマンドで表示し，そのあとでdumpコマンドでダンプするということになります。ところで，forをバッチファイルで使う場合，変数名に「%数字」は使うことができません。この理由はあとで説明します。

●echo〔on/off/《文字列》〕

文字表示を禁止したり許可したりするのに使います。また，「echo《文字列》」を使えば，文字表示が禁止されていてもメッセージを表示することができます。バッチファイルでファイルをコピーする場合，

```
A>copy ～ ～
```

などという表示が何行も出るのはみっともないので，通常バッチファイルの先頭で，

```
echo off
```

と指定し，余計な文字の表示を禁止します。

しかし，なにも出ないのも寂しいので，

echo お待ちください

などと表示させたりもします。

●**pause〔《文字列》〕**

これはバッチファイルの実行を一時中断する命令です。ask68kを使うためにconfig.sysを書き換えてリセットした場合などは，RAMディスクは保存されていますので再びファイルをRAMディスクに転送する必要はありません。そこでバッチファイルを，

pause ファイルを転送します
copy 〜

のようにしておきます。これで，画面に「ファイルを転送します」と表示された時点でバッチファイルの実行が一時中断されるようになります。ファイルを転送しないならCTRLキーを押しながらCを押せば，

バッチ処理を中止しますか <Y/N>

と表示されますので，Yキーを押して無駄な転送を省くことができるというわけです。ファイルを転送するならCTRL+C以外のキーを押せばOKです。また，pauseの後ろに文字列を書かなければ，

何かキーを押してください

と表示され，実行が中断されます。ここでもCTRL+Cを押すことで，バッチファイルの実行を一時中断することができます。

●**if〔not〕条件　コマンド**

条件を判断します。BASICやCのように，大小関係などを判定するのではなく，特定の条件の判定に限られています。指定できる主な条件を挙げておきます。

・**《文字列》==《文字列》**

文字列が等しいかどうかを調べる。等しければコマンドが実行される

・**errorlevel《数値》**

エラーコードが数値以上ならコマンドを実行する

・**exitcode《数値》**

エラーコードが数値と等しければコマンドを実行する

・**exist《ファイル名》**

《ファイル名》のファイルが存在すればコマンドを実行する。カレントディレクトリに存在するかどうかを調べるので，ほかのディレクトリにあるかどうかを知りたければパス名付きでファイルを指定すること。

ところで，errorlevelとexitcodeのところで「エラーコード」という言葉が出てきました。これを説明しておきましょう。コマンドやファイルを実行すると，その実行が正常に終了したかどうかを示す数値が返ってきます。これがエラーコードです。通常のコマンド実行ではこれらのエラーコードを見ることはできません。例として，異常があった場合に1を返すプログラムがあるとしましょう。名前を仮にcmnd.xとしておきます。バッチファイルで，

cmnd
if exitcode 1 echo "エラーです"

とすれば，エラーが発生した場合のみ「エラーです」というメッセージが表示されることになります。もっとも，一般にプログラムはエラーが発生した場合はメッセージを表示しますので，実際には次に説明するgotoコマンドと併せて使用するのが普通です。existは，RAMディスクに辞書がなければ転送するなどの用途に便利です。

if not exist x68k_m.dic copy 〜
if not exist x68k_s.dic copy 〜

というバッチファイルをRAMディスクで実行すれば，必要な場合のみコピーできます。

●**rem〔《コメント》〕**

これ以降のコマンドはバッチファイル中で実行することを目的に作られていますので，コマンドラインで実行しても無意味です。remで始まる行はバッチファイルでは無視されます。そこでremはバッチファイル中にコメントを入れるのに使用されます。また，「echo on」の場合，画面には，

A>rem 辞書コピー中

とREMがそのまま残りますから，メッセージ表示に使うこともできます。

●**goto《ラベル》**

バッチファイル中で処理の流れを変えるのに使います。ラベルは，

:label

のように：で始まる行です。辞書をRAMディスクにコピーする例をgotoコマンドを使って書くと以下のようになります。

rem C はRAMディスク
if exist c:x68k_m.dic goto end
copy b:¥dic¥x68k_m.dic c:
copy b:¥dic¥x68k_s.dic c:
:end

辞書がすでに存在するなら，コピーを飛ばしてラベルendにジャンプするわけです。

●**shift**

バッチファイルはパラメータを与えて呼び出すことができます。shiftはこのパラメータをシフトする命令です。詳しくは次の章で説明します。

バッチとパラメータ

Human68k ver2.0のformat.xは，単に起動したらメニュー画面になりますがformat b:のようにフォーマットするドライブ名をパラメータとして与えることもできます。またed.xもパラメータとして編集するファイル名を与えることができます。

バッチファイルもこれらのコマンドと同じようにパラメータを受け取ることができます。パラメータは1文字の数字の前に%を付けて表します。%1は1番目のパラメータ，%9は9番目のパラメータを意味します。

簡単な例をお見せしましょう。

echo off
echo %1
echo %2
echo on

というバッチファイルtest.batを作り，

test ABC DEF GHI

とすれば，画面には「ABC」と「DEF」が表示されます。forの変数に%数字を使えないといったのは，このバッチファイルに与えられた引数と区別できなくなるからです。というより，%数字はその行が実行される前に，与えられた引数によって置き換えられてしまうのです。先のtest.batの1行目，「echo off」を削除して，

test ABC DEF

としてみてください。画面には，

A>echo ABC
ABC
A>echo DEF
DEF
A>echo on

と表示されたはずです。「echo %1」が「echo ABC」に置換されていますね。もし，

for %1 in (*.doc) do type %1

という行がTEST.BATの中にあれば，

for ABC in (*.doc) do type ABC

となってしまい，エラーになってしまうわけです。

バッチファイルのパラメータは%に続く1文字の数字です。しかし，%0はバッチファイル名そのものなので実際に利用できるのは%1〜%9の9個ということになります。では，もっと多くのパラメータを取

りたい場合にはどうすればいいでしょう。また，複数個のファイルを一気に削除してくれるdels.batというバッチファイルを作ろうと思ったときには，

```
del %1
del %2
 ⋮
del %9
```

と何行も書かなければいけないのでしょうか。ここで登場するのがshiftコマンドです。これはパラメータをひとつずらす命令で，%1を%0へ，%2を%1へとひとつずつずらしてくれます。%9にはいままで参照できなかった10番目のパラメータが入りますので，これでパラメータがいくつあっても大丈夫。また，複数のファイルを削除するバッチファイルも，%1を削除してshiftするという動作を繰り返せば，対処できます。しかし，ループ回数を指定する命令はないので，

```
"%1" == ""
```

のようにパラメータが尽きたということを終了の条件にする必要があります。

バッチ処理では%1と書けばそれが文字列の中だろうがおかまいなしに与えられたパラメータに置き換えられます。純粋に%1という文字列を表現したければ，%%1とする必要があるのです。よって，先のforも，

```
for %%1 in ～ do type %%1
```

とすればバッチ処理の中で使えるようになります。もっともわざわざこんなことをしてまでforの変数に%数字を使いたいなんて人は少ないかもしれませんが。以上のことを使うと，dels.batは次のようになります。

```
:continue
if "%1" == "" goto end
del %1
shift
goto continue
:end
```

バッチと環境変数

環境変数はバッチ処理における変数のように使うことができます。環境変数に値をセットするには，command.xでやるのと同じようにsetを使えばOKです。参照するときは環境変数名を%と%で囲みます。簡単なテストをしてみましょう。

```
set test=abc
echo %test%
```

という内容のバッチファイルを実行すると，

```
abc
```

と表示されます。では，複数のファイルを特定のディレクトリにコピーするcp.batを作ってみましょう。

```
echo off
set tmp=%1
:continue
if "%2" == "" goto end
copy %2 %tmp%
shift
goto continue
:end
echo on
```

cp.batは1番目の引数にコピーしたいディレクトリを，2番目以降の引数にコピーしたいファイル名を列挙して使います。shiftによってパラメータがずれてしまいますので，最初に%1を環境変数にコピーしておいてから，ひとつずつ順番にコピーしていくようになっています。

●RAMディスクにファイルをコピーしたい

先日，友人宅のX68000で，自分のシステムディスクをドライブに入れ，RAMディスクにファイルをコピーしようとしたときの話です。リセットしたあとRAMディスクが確保され，ファイルの転送が始まると，なんと，ハードディスクのアクセスランプが点滅しているではありませんか。友人のマシンのCドライブはハードディスクだったのです。

転送したのはcommand.xとエディタのみでしたので，該当ファイルを削除するだけで修復でき，大惨事は免れました。もし，ルートディレクトリに辞書が置いてあって，自分の辞書で上書きしてしまったら……，と思うとゾッとしました。

そこで作ったのがRAMディスクを自動判別するコマンドdrv.xです（リスト1）。とりあえずバッチコマンドとHumanの標準コマンドだけではできないのでC言語で作ってみました。また，リスト2にはdrv.xを利用したautoexec.batのサンプルを掲載します。drv.xは環境変数ramdiskに最初のRAMディスクドライブ名をセットします。もし，RAMディスクが存在しない場合は，画面にメッセージ表示し，エラー番号1を返します。

リスト2のバッチファイルはエラー番号が1ならなにもせず終了，そうでなければpauseします。ifのあとの「%ramdisk%」という行で，カレントドライブをRAMディスクに移しています。環境変数ramdiskには「C:」のようにドライブ名に：を付けたものがセットされるので，ドライブの移動ができるわけです。続いてファイルを転送します。

●バッチからバッチを呼びたい

バッチファイルを実行したときに画面に表示される文字をリダイレクトして再利用したい。そう思って，

```
test.bat > temp.doc
```

などとやっても，temp.docにはなにも入りません。バッチ処理中に画面に表示された文字は，そのままではリダイレクトできないのです。こんなときには，

```
command test.bat > temp.doc
```

と，command.xのパラメータとして実行しようとしているバッチファイルを指定してやればOKです。

この方法はバッチファイルの中から別のバッチファイルを呼び出して使いたい場合にも使えます。以下のようなバッチファイルtest.batの場合を考えてみましょう。

```
echo off
test1.bat
test2.bat
```

リスト1　drv.c

```
 1: /* ｺﾝﾊﾟｲﾙ方法 : cc /Y drv.c */
 2: #include <stdio.h>
 3: #include <doslib.h>
 4: struct DPBPTR DPB;
 5: void main()
 6: {
 7:     int  flag
 8:     int  i;
 9:     char drv[] = "A:";
10:
11:     for ( i=1; i<10; i++ ) {
12:         flag = GETDPB( i, &DPB );
13:         if ( flag < 0 ) {
14:             fprintf( stderr, "RamDiskはありませ
 ん¥n" );
15:             exit( 1 );
16:         }
17:         if ( DPB.id == 0xF9 ) {
18:             drv[0] = 'A'-1 + i;
19:             printf( "RamDiskは%sです¥n", drv );
20:             SETENV( "ramdisk", 0, drv );
21:             break;
22:         }
23:     }
24: }
```

リスト2　autoexec.bat

```
 1: PATH=A:¥;A:¥BIN;A:¥SYS;A:¥BASIC2
 2: echo off
 3: drv
 4: if ERRORLEVEL 1 goto end
 5: %ramdisk%
 6: pause ファイルを転送します
 7: copy a:¥command.x
 8:      ……
 9: :end
10: echo on
```

```
echo on
```

このように単にバッチファイル名だけを書いたのでは，test1.batが実行された時点でバッチファイルの処理が終了してしまい，test.batの続きは実行されませんが，

```
echo off
command test1.bat
command test2.bat
echo on
```

と，すれば，test1.batを実行したあと再びtest.batに制御が戻り，次にtest2.batを呼び出すことができるようになります。

バッチでメニュー選択

ワープロを使うときは辞書をRAMディスクに転送したい。Cコンパイラを使うときにはインクルードファイルとライブラリをRAMディスクに転送したい。このように，使うアプリケーションに合わせて，少ないRAMディスクを有効に使いたいというのは誰もが考えることでしょう。

ところが人間というやつはどうしようもないほどなまけもので，そのうちバッチファイル名を打ち込むのさえ面倒だと思うようになります。起動するアプリケーションの一覧を表示しておき，番号を入力するだけで必要な処理を全部やってくれたら……。

そこで，バッチファイルの中でキーを入力するいい方法がないかとマニュアルを眺めてみたのですが，どうも見当たりません。よって，またプログラムを組みました。

リスト3はそんなものぐさなあなたに贈るinput.xコマンドです。この，C言語のプログラムは，キー入力した文字列を環境変数inputにセットするだけの簡単なものです。このinput.xコマンドを利用したらバッチファイルから番号の入力ができるようになります。その，実際の応用例がリスト4です。リスト4はechoでメニューを画面に表示し，キー入力を待ちます。3行目と7行目は，実はechoに続いて半角の空白が2つ入れてあります。空白を表示しても画面にはなんの変化もありませんから，画面上ではあたかも改行したように見えます。単にechoとだけ書いたのでは，「echoは<on>です」と表示されてしまうので注意。改行を表示する常套手段として覚えておきましょう。

input.xが終了すると%input%にセットされている文字と番号を順に比較し，それぞれの選択肢へgotoし必要な処理を行います。それぞれの目的別のバッチファイルを作っておき，その名前を書いておけばいいでしょう。ここは皆さんの工夫次第というところでしょうか。非常に簡単なプログラムながら，input.xは重宝なコマンドです。

リスト3 input.c

```
 1: /* ｺﾝﾊﾟｲﾙ方法 : cc /Y input.c  */
 2: #include <stdio.h>
 3: #include <doslib.h>
 4: void main()
 5: {
 6:     int  ch;
 7:     char input[100], *ip = input;
 8:
 9:     while (( ch=getchar())!='¥n' )
10:        *ip++ = ch;
11:     *ip = 0;
12:     SETENV( "input", 0, input );
13: }
```

リスト4 run.bat

```
 1: echo off
 2: echo どのプログラムを起動しますか
 3: echo
 4: echo 1)  ワープロ
 5: echo 2)  DEFSPTOOL
 6: echo 3)  Cコンパイラ
 7: echo
 8: rem キー入力開始
 9: input
10: if %input% == 1 goto wp
11: if %input% == 2 goto defsp
12: if %input% == 3 goto cc
13: goto end
14: :wp
15:   ワープロ用環境設定・起動
16: goto end
17: :defsp
18:   DEFSPTOOL.BAS用環境設定・起動
19: goto end
20: :cc
21:   Cコンパイラ用環境設定・起動
22: :end
```

リスト5 welcome.bat

```
1:  echo ようこそX68000の世界へ
2:  copy x68000.opm opm
3:  prompt $e[37m:$d:$t^[E$l:$p:$g:$e[33m
```

リスト6 cl.bat

```
 1: echo off
 2: :START
 3:   shift
 4:   if "%0" == "" goto END
 5:   if not exist %0.c goto ERR1
 6:   echo %0.cのコンパイル開始
 7:   cc %0.c /E > nul
 8:   if errorlevel 1 goto ERR2
 9:   goto START
10: :ERR1
11:   echo %0.c が見つかりません
12:   goto START
13: :ERR2
14:   echo エラー%0.errを見て下さい
15:   goto START
16: :END
17: echo on
```

リスト7 login.bat

```
1: echo off
2: if "%1" == "" goto ERR
3:   set usr = %1
4:   cd %usr%
5:   echo ここは%usr%君の世界！！
6: :ERR
7:   echo 使用方法: login ﾕｰｻﾞｰ名
8: :END
```

番号を入力してもらうときだけでなく，文字列を受け取りたい場合なども使えることと思います。ぜひ活用してみてください。

バッチり発展学習

では，最後に，バッチファイルのサンプルをいろいろ作ってみましょう。

例1：まず，「ようこそX68000の世界へ」とメッセージを出し，音楽「X68000のテーマ」を鳴らし，ついでにプロンプトを変えるバッチファイルwellcome.bat（リスト5）。これなんか，autoexec.batに書いておけば元気よく仕事が始められそうですね。ところで，このリストで使われているx68000.opmという名前のファイルは，自分で作らなければいけません。作り方は今月の質問箱を参考にしてください。また，promptのあとの長い，一見わけがわからない文字列は，それぞれ意味があります（$dは日付表示で，$e[33mは画面属性のリセットなど）がここでは解説しません。Human68kのマニュアルのpromptのところに載っているので見てください（^[は^V+^[です)。

例2：ちょっとは実用的なCコンパイル用バッチファイルcl.bat（リスト6）。これは，コンパイルしたいファイル名リストをパラメータとして渡して，コンパイルしていくもの。コンパイルエラーは拡張子がerrのファイルに取られます。また，ファイルがなかったりコンパイルエラーが発生したらメッセージを出し，次のファイルをコンパイルします。

例3：大容量のハードディスクを複数の人で使用する場合に便利なlogin.bat（リスト7）。まず，ユーザー名をパラメータとして渡し，それぞれの名前を環境変数usrにセットして，その名前が付いたディレクトリに移動します。これなど，親子兄弟でX68000を共有している清く正しい家庭には欲しいですね。しかし，「母さん俺の隠しファイル読んだだろう」などと家庭不和の原因になっても知りません。

このように，バッチファイルではさまざまなことができます。これはもう立派なプログラミング言語であるといっても過言ではないでしょう。皆さんもいろいろと面白そうなバッチファイルを考えてみてください。そして便利なバッチファイルができたら，投稿してください。

コマンドに歴史あり

マジカル・ヒストリー・ツアー

Ogikubo Kei
萩窪 圭

使いこなすことができれば，コマンドラインの環境を飛躍的に改善できるのに，あまりにも高機能なためか活用されずにいることが多いHuman68k v2.0のヒストリドライバ。ここでは驚異のヒストリパワーを垣間見てみましょう。

なんの間違いか神の試練か知らないが，世間でいちばん多くの人が買ってしまうパソコンでは，キーボード操作でディスクイジェクトができないらしい。それどころか，別売のMS-DOSのコマンドシェルでは，1回前のキー操作しか覚えていないらしい。だから，私はその日本でいちばん多くの人が使っているパソコンを前にするといつもストレスが溜まる。確かに，PDSとかでずっと昔のキー操作まで覚えてくれるツールもあるらしい。しかし，それはそういったパソコンを買ってしまった人たちの苦肉の策であり，苦肉があることを知らない大多数の人は，ひとつ前の入力しか覚えていないのを当然と思っているに違いない。

Human68kのおかげです

X68000となるとそうは問屋の大安売りである。なんてったって，昔入力したコマンドを覚えているなんて常識だから。それどころか，Human68kの新しいバージョンでは，やたらめったらたくさんの機能がついてしまった。機能が増えると，それは覚えることが多くなったりするわけであり，作り方が悪かったりすると，古いバージョンとキー操作が違ってしまったりするわけで，あまりフレンドリーじゃなかったりするのだけれど，そこはそれ，最初から全部覚えようとするからいけないのだ。

そういったわけで，もう1980年代も終わってしまう（しまった）のである。あと11年で21世紀になってしまうのである。21世紀といえば月でモノリスが発見されてしまう年である。ストローの中の液体が下がってしまうのは，ストローの刺さっているパックの内圧と外圧の差から，パックの膨らもうとする力で一緒に液体も吸い込まれたわけであるから，正しいのである。

というわけで，history.xも正しく使えば正しいのだ。

history.xの2つの顔

history.xには2つの顔がある。ひとつはconfig.sysの中に隠れているデバイスドライバの顔。もうひとつはコマンド行から実行できるコマンドとしての顔だ。これがまた話をややこしくしている。なぜなら，history.xには起動時オプションというやつが全部で66個もあるからだ。ぎょえーっというようなもんである。しかも，起動時にのみ有効なもの，いつでもOKなもの，起動時には使えないものの3種類あるのだ。私などはもの覚えの悪さが自慢で，66個中1個しか記憶にない。

ここで大事なのは，1個しか知らなくても，なんの不自由もないということである。66個あるといったって，よく似た機能のものを一緒くたにしたりデフォルトでかまわないものを省いたりすればぐっと数が減るのだ。そのうえ，オンラインヘルプが（一応！）ついているし，必要だと思ったオプションは（起動時のみ設定というやつは駄目だけれど）その都度マニュアルを見ながらエイリアスに登録してしまえばいいだけのことだからだ。エイリアスについてはあまりにも気に入ったので，別記事を書いてしまった。詳しくはそっちを読んでください。読めばすぐわかるようになっている（はず）。

とりあえず，どんなものがあるかだけは把握しておこう。それがhistory.xの機能に直結しているからだ。

・初期設定・機能設定
・コマンドラインでの編集機能
・ヒストリ機能の設定
・エイリアス機能の設定
・簡易バッチ機能の設定
・ディレクトリ変更履歴機能の設定
・ヘルプやヒストリを記録したファイルの入出力機能
・バッファクリアの機能

と，まあこんな感じ。こうして分類するとそれほどでもないが，起動時オプション

ヒストリ機能の心

ヒストリー＝歴史（と，教えられてきた我々）である。history.xも歴史である。過去の行いを無駄にしないよう精進するから歴史なのである。過去の行いを繰り返したくない，それがコンセプトなのだ。そして，あまりフレンドリとはいえないコマンドシェルにおいて少しでもユーザーのストレスをためないためには，無駄にキーを押させないのがとりあえず'80年代の解決なのだ。

基本はVer.1でもお馴染みの，とりあえず「昔使ったコマンドは覚えていられるだけ覚えてるから，いつでも使ってちょ」の姿勢だ。こいつが発展して，command.xから独立したデバイスドライバになったのがhistory.xと思ってよい（もちろん，command.xだってお馴染みの機能は持っているから，history.xを使わなくても，Ver.1の環境は得られるよ）。

では，history.xの機能を見てやろう。

history.xはデバイスドライバであるから，かな漢のASKみたいに常駐する。それでもって，ユーザーが何かキーを押すと，command.xがそれを解釈するのだが，その前にちょっと横取りして，history.xのためのキー入力だったら，そいつを実行してしまうのだ。たとえば，command.xでは[HELP]キーは“his”，つまりヒストリに登録されているユーザーの所業をずらずらっと見せてくれるのが役割なのだが，history.xがいると，command.xが“his”する前に，history.xにおける[HELP]キーの役割，オンラインヘルプを先に実行してしまうのだ。これはまれな例で，たいていはcommand.xで使わないキーをhistory.xの機能発揮に割り当てている。だから，[ESC]-うんちゃら，とか[CTRL]+かんちゃらといった覚えにくい羽目に陥ってしまうのだね。で，どうすれば使いやすいかというとやはりマウスかなんかでピッとやれればいいんだろうけれど，それではキーボードでちょんちょんとやりたい人向きではないし，マウスでGUI（なんのことかと思ったら，グラフィックユーザーインタフェイスの略だそうだ）すると，絵を描いたりなんだかんだでプログラムサイズは大きくなるわメモリは食うわの大騒ぎとなってしまう。'90年代には“メモリ食っても心は錦”の精神が普及するかもしれない。

だけ知っていればいいものではない。とりあえず順番に見ていくことにする。

デバイスドライバの顔

組み込み時のみに使えるオプションはこの部類だ。組み込み時ということは，config.sysの中に書くときのオプションということである。よって，頻繁に書き換えたりはしないものである（はず)。だから，覚える必要はない。

まず，history.xの4大機能といえるエイリアス，簡易バッチ，ヒストリ，ディレクトリ変更履歴で使用するバッファの大きさを指定する。デフォルトではそれぞれ2Kバイトである。大きくする必要はそれほどないだろう。

それから，“エイリアスの中でエイリアスを使うかどうか”や，ヒストリの内容を保存するファイル（ヒストリ定義ファイル）やキー定義ファイル（key.sysとは異なったファンクションキーを定義できるのだ）の場所を指し示すなどがある。ファイルの場所はhistory.xと同じディレクトリがデフォルトなので，そういった使い方をしている限り気にする必要はない。しかし，デバイスドライバはみんなディレクトリ¥sys¥にして，ヒストリ関係のファイルは¥his¥だな，てことになると，“/d¥his¥”(history.hisとkey.hisの場所を示す）なんてすることになる。

起動時専用オプションでなくとも，1回設定してしまったら変えることはないや的なものはconfig.sysに入れてしまうべきである。コマンドライン編集時に挿入モードにするか上書きモードにするかとか，ヘルプファイルの位置指定とか，エイリアスは使うけどディレクトリ変更履歴は名前が長いから使わないというときにディレクトリ変更機能を殺してしまうとか，だ。

たいていのオプションはいつだって使えるものなので，変な設定をしてしまったとしても，あわてることはあるまい。

このようにしてhistory.xはデバイスドライバとしてシステムの一部となる。

例）history関係のファイルを全部A:¥his¥に置き，起動時上書きモード(デフォルトは挿入だ)でなおかつ挿入/上書きを変更したときにその状態を保存するようにし，ディレクトリ変更履歴機能を使わないようにするとき。

```
device=history.x /da:¥his¥
/va:¥his¥ /ar /o /nc
```

Human68kのコマンドとしての顔

config.sysに組み込んで立ち上げると，コマンドシェル環境にカーソルがある限り，history.xは否応なく働いてくれる。が，それとは別に，history.xをコマンドとして実行するといろいろとヒストリ関係の力を発揮する。たとえば，opmdrv.xはFM音源ドライバだけど，opmdrv offとするとopmdrv.xを無視する(こうすると，少しだけ処理速度が速くなるのだ）ようなものだ。

コマンドとしては66個のうち，60個が使える。うーん，多いね。historyとだけ打って実行するとずらっと約4画面分のメッセージが出てくるくらいだ。

このうち各種機能用はその都度説明することにするとして，まずはバッファクリアである。あんまり使わないと思うけど，溜まったヒストリやエイリアスやディレクトリ変更履歴などをえいやっとクリアしてしまうのだ。

続いて，バッファの内容をファイルに書き出したり，ファイルから読み込んだりする機能だ。こいつはおいしい。history.xの世話になったキー入力を全部とっておけるので，電源を切ってもリセットしても溜め込んだコマンドたちを無駄にしなくてもすむのだ。

バッファの内容をファイルに書き出すには/w系のオプションを用いる。/w系はさらに/wnと/w+nに分かれる。+は追加するときの記号で，+がないと，ファイル内のデータをバッファ内のデータに置き換えるわけだ。ここでnにはA, B, C, H, @の5種類が入る。Aはエイリアスの A，Bは簡易バッチのバッチのB，Cはディレクトリ変更履歴（つまり，CDコマンドの履歴）のC，HはヒストリのH，@は全部いっぺんに書き込んでしまうことだ。

まあ，ヒストリなんて全部書き込んでしまうととても長くなってしまうのが常だが，ヒストリ定義ファイルは普通のテキストファイルなので，エディタで不要なものや無駄なものを（打ち間違いや同じコマンドを何度も使ったところ）削ってやれば，結構少なくなるはずだ。1回やってみるといいぞ。いかに無駄な入力や繰り返し入力が多いか気づくはずだ。

逆に，ファイルからバッファに読み込むときは/r系のオプションを使う。あとは/w系と同じだ。

例）エイリアスだけファイル（Aドライブのディレクトリhisのhistory.his）に書き込む

```
history /waa:¥his¥history.his
```

ヒストリバッファをクリアする

```
history /ch
```

では各種機能の説明だい!

最初は基本中の基本であるヒストリである。command.xの持つヒストリとだいたい同じなのだが，テンプレートが[ESC]コマンドでも実行できるようになったこと，command.xのヒストリでは一度ヒストリから引っ張ってきたコマンドをテンプレートに入れてからコマンド行へ呼ぶ方式だったが，history.xでは直接入力行に呼び出せるようになったことなどが大きな進歩。つまり，テンプレートを介さずにヒストリが実行できるのだ。そのうえテンプレートも[ESC]コマンドでOKということは，ファンクションキーを使わなくてもよくなったということで，遠いところにあるファンクションキーが大嫌いな私としてはとてもうれしい。

また，テンプレートには直前の入力が納まっているわけだが，テンプレートに納まったn番目のパラメータだけをカーソル位置にコピーという技までついたのだ。

例）type config.sysのあとに，ed config.sysをしたいとき

```
ed [ESC]-1
```

と打つと，

```
ed config.sys
```

となる。

めでたいめでたい。コマンドだけでなくパラメータの2度打ちも少しは避けられるようになったんだね。

進歩したのはテンプレートだけではない。ある文字列から始まるコマンドをヒストリバッファから検索するとか，ある文字列を含んだコマンドを検索するといった，ワープロの検索機能並みの技がついた。さらに，任意の文字列を含むコマンドのリストを表示するなんてのもある。そのうえ，パラメ

ータだけを表示したり検索したりもできるのだ。うむうむ。こういった機能は使ってない人が多いだろうなあ。だって，覚えられそうにないんだもん。

だがしかし，よくマニュアルを見るとたいていパターンが決まっている。試験前の英単語馬鹿覚えよりはずっと簡単だ。

とりあえず，列挙する。

CTRL＋W系 ROLLDOWNと一緒
CTRL＋E系 ROLLUPと一緒
CTRL＋R系 パラメータ相手のROLLDOWN
CTRL＋T系 パラメータ相手のROLLUP
CTRL＋L系 リストのLだ。1画面分のヒストリのリストを表示
ESC-L系 ヒストリ全部のリスト表示
ESC-^ パラメータ部分のROLLDOWN
ESC-¥ パラメータ部分のROLLUP
ESC-N 数字を入れてESC-Nすると，その番号のヒストリ内容を表示

なんとかなりそうだね。WERTのキーはキーボードの上段に並んでいるし。

さて，パラメータとパラメータ部分の違いだが，パラメータというのはひとつひとつのパラメータを指し，パラメータ部分というのは，ひとつのコマンドに対するパラメータ全部を指すことをチェック。

続いて，〜系とあるのは，CTRL＋n(nはWERTLのどれか）を打つ前に文字列を入力しておくとその検索をしてくれると思ってよい。文字列＋スペースだとまた意味が違うけれど，まあ，細かいことだな。

異色な機能としては，

ESC-@

がある。これは，入力行の文字列をヒストリに放り込み，入力行を削除（入力行からヒストリへのムーブと思えばいい）だ。実行しなくてもヒストリに登録できる技が使えるのだ。

なお，これらで検索しても，その結果は入力行に反映されるだけで，テンプレートには入らないことに注意されたい。−や＋は従来どおりテンプレートへの検索としてcommand.xのヒストリ機能が生きているから，それを使おう。

前後にカーソルが動くんだぜ

Human68kのバージョン2を使って最初に驚いたのは，“コマンド行でカーソルが自由に行き来できる！”ということだった。DOSではカーソル移動はできないのが常識だったからね（カーソルキーでカーソルを戻そうとしたらバックスペースしちゃって困った経験は誰にでもあるはずだ）。

それがなくなった。コマンド行が1行エディタのように使えるようになったのだ。だから，7文字前の打ち間違いのために7文字も打ち直すことはしなくてもよい。ed.xの持っている1行対象の機能はだいたい使えると思ってよいだろう。

カーソルの前後移動（CTRL＋S，CTRL＋D，→，←)，1ワード移動（CTRL＋F，CTRL＋A)，行頭/行末への移動(CTRL＋B)，カーソル位置の文字削除（CTRL＋G，DEL)，もちろんバックスペース，行末まで削除（CTRL＋K)，行頭まで削除(CTRL＋U)，大文字/小文字切り替え（CTRL＋])，コントロールコードの入力（CTRL＋[，ESC）など。1ワード削除がED.xと違っている（CTRL＋^，CTRL＋¥）のは例外だ。

CTRL＋Sがスクロールの一時停止と重なっているけれど，それはそれで働いてくれるので問題はない。

問題なのは，挿入/上書きを切り替えてもINSキーのランプがついたり消えたりしてくれないことだ。困ったものだ。

なお，少々特異な機能として，環境変数の展開がある。環境変数は%環境変数%とすると使えるのだが，そこでESC-/を入力すると，環境変数をその内容に置き換えてくれる。

cdコマンドはよく使う

Human68k（MS-DOSでもいいけど）で最もよく使うコマンドは？　と聞かれた場合，みんなはどれを思い浮かべるだろうか。編集部でその話題が出たとき，優勢だったのは“cd”コマンドだった。たいてい，cdかdirだろう。そこで，history.xではcdコマンドだけ，ヒストリバッファとcd専用バッファの両方に放り込む。cdコマンド専用バッファ機能を面倒臭くいうと，ディレクトリ変更履歴機能となるのである。

で，cdコマンドだけ集めた普通のヒストリだと思ってくれてよい。使うのは，

CTRL＋Q系 cd相手のROLLDOWN
CTRL＋Y系 cd相手のROLLUP
CTRL＋_系 cd履歴のリスト表示

QとYはヒストリで使うWERTの両側にあるので，覚えやすいだろう。〜系とあるのは，検索なんかもできるからだ。パスの入力が面倒だという人はよく使う全ディスクの全ディレクトリをhistory.hisに登録しておいて，CTRL＋Yかなんかで呼び出せば，きっと少しは楽になるだろう。

結構めんどうな簡易バッチ

簡易バッチ機能というのは，ヒストリに溜まったコマンドから一部を抜き出して，登録してしまおうという機能だ。たとえば，Cでプログラムなんか作っていると，edコマンドとccコマンドと実行の繰り返しだなあ，ということに気が付いたりする（よくCを使う人ならとっくにバッチファイルにでもしてるか)。そのとき，ヒストリのリストをCTRL＋Lで見て，

{1} ed pgm.c
{2} cc /y pgm.c
{3} pgn
{4} pgm

を見つけたとすると（{3} はpgmと打とうと思って，タイプミスした)，

1-2,4 CTRL＋N

と入力すると，1と2と4が簡易バッチバッファに登録されるのだ。次からはCTRL＋Nと打つだけで(実行するかどうか聞いてくるところがかわいい)，実行してくれる。また，中断待ち時間の設定というのがhistory.xのオプションにあって，これを1秒にすると，1行実行するたびに1秒待ってくれるので，その間に中断できる。

なお，ヒストリひとつにつき簡易バッチひとつしか登録できないのでご了承を。

command.xのヒストリとの違い

history.xでは複数のヒストリを持つことができる。と，マニュアルにはあったんだけど，どうもよくわからないので調べてみた結果，次のような事実が判明した（んな大げさなもんじゃあないけど)。

command.xのヒストリは（当たり前だけれど)，command.x1回分しか持たない。たとえば，チャイルドプロセスでcommand.xを起動したり，commandとプロンプトから入力して起動したりすると，そのcommandのヒストリとなり，せっかく

図1　ヘルプファイルの構造

```
        :
  <テキスト>
        :
 ##M<メニュー>←―何でもよい
 ##S<押したキー>=<ページ数>…
 ##PE

 ##E              ←―終了のマーク
 ##P<ページ名>
        :
  <テキスト>
        :
 ##M<メニュー>
 ##S<押したキー>=<ページ数>…
        :

 ex)
 ##S1=1␣^C=E␣^[=1  1を入力，あるいは
  :                 ESCでページ1へ
  :                 CTRL+Cでエンド
 ##P1
  :
```

溜まったヒストリは使えないのだ。しかし，history.xのヒストリはいつでもどこでも有効なのである。チャイルドプロセスしようがどうしようが，一度溜めたヒストリは使えてしまう。

たとえば，コマンドシェルからcommandと入力して新しいプロセスに移ったとき，hisコマンド（これはcommand.xのコマンドだ）を実行しても，

　00000:his

となるだけだが，[ESC]-Lでヒストリのリストを表示してみると，電源を入れてからの全ヒストリが表示されるはずだ。

しかし，である。history.xのオプション/bpを実行すると，プロセスによって使うヒストリが変わるのだ。また，/bbオプションだと，次の番号のヒストリに切り替わる。これはおもしろい。

たとえば，

　history /bb

をする。ここで[CTRL]+Lすると，表示されるヒストリが“ヒストリ{2}履歴”となるはずだ。つまり，別の歴史に（パラレルワールド）入り込んだかのように，いままでの入力をなかったことにする世界が始まるのだ。history.hisに2番目のヒストリとしてcc関係を集めようと思ったとする。

　cc history /b | | cc %1 %2 %3 %4 %5 | | history /b

というのがエイリアスに登録してあったとしよう。ccのあとに%1から%5まであるのは，このくらいあれば大丈夫かなという安易な気持ちの表れ。で，history /bでヒストリが切り替わるから，これでコンパイルすると，コンパイルコマンドだけヒストリの2に溜まっていくのだ。簡易バッチもヒストリひとつに対してひとつだから，ヒストリを複数用意すると，簡易バッチも複数できる。用意するヒストリの数はconfig.sysの中だけでしか設定できないので注意。

ほかにも使い方がある気がするけどなあ。どうもよくわからない機能なので，すみません，と謝ってごまかしておこう。

それから，history.xとcommand.xのヒストリの違いだが，こんなのもある。

command.xのヒストリは入力したコマンドをとにかく全部溜め込んでいくが，history.xのヒストリは，“同じコマンドを何回も連続して実行しても，ひとつしか溜め込まない”のである。つまり，無駄な歴史を残さないわけね。

ついでに，key.sysとkey.hisの違いも無視してはいけない。なぜなら，ファンクションキーやら各種特殊キーの機能がkey.sysとkey.hisで若干異なるからだ。

key.sysとkey.hisでは，前者がcommand.xの，後者がhistory.xのファンクションキー定義ファイルであることを確認しておく。history.xを組み込んでいるときはkey.sysではなくkey.hisが使われているのだ。

両者の違いは，[HELP]キーやカーソルキーなどに見て取ることができる。ここで，次のような人が現れるわけだ。

「エイリアスやコマンドライン編集機能はおいしいけど，カーソルキーなど編集キーの機能が変わってしまったのはけしからん」

そういう人の対処は2つある。ひとつはそういうときに，history.xの機能を殺してしまうことである。

　history /k

で，ヒストリは陰に隠れ，ないものとして使えるのだ。ヒストリの復活は，

　history /u

である。これで，古い環境でも違和感なく使える。

で，もうひとつ，編集キーをバージョン1と同じにする方法がある。

key.xを使って，key.hisを書き換えればいいのだ。key.sysとkey.hisをまったく同じにしたければ，key.sysをkey.hisにコピーしてやればいい。構造は同じだから可能だ。そうすると，ROLLUP/ROLLDOWNや[HELP]などは昔のままで，新しい機能は[CTRL]や[ESC]を使って実行できる。

また，マニュアルには[CTRL]+OとINSキーで挿入/上書き切り替えと嘘が書いてあるが，key.xでkey.hisのINSキーを[CTRL]+Oにすれば嘘ではなくなる。

これで，ヒストリは完璧だ。ん？　まだあった。あの驚異的な遅さを誇るヘルプ機能だ。

ヘルプ機能のおいしい使い方

history.xではhistory.hepなるヘルプファイルを用意してあって，[HELP]キー（あるいは[CTRL]+V）で呼べるようになっている。

これは，メニュー選択式に目的のヘルプ画面へ行けるようになっているものだが，メニューからメニューへいくのがひどく遅いのだ。ディスクを読んでいるわけではないのに，である。そこで，ヘルプファイルを書き換えて，短くしてしまえばよいのではないかと思い立った（短ければそんなに待たされない）。ヒストリに限らない自分だけのヘルプだって作れるのだ。

ヘルプファイルは図1のような構造になっているから，エディタで簡単に作れる。注意すべきは，1ページ1画面に収めようと思ったら，行数を数える（足りないところは空白行を補う）必要があることか。

＊　＊　＊

知らなければ知らないですんでしまうのが世の中の怖いところで，エイリアスしか使っていなかったとか，コマンドライン編集しか知らないといった人は多いと思う。しかし，知ってしまったからには，もう戻れない。特に自分が楽できることには。

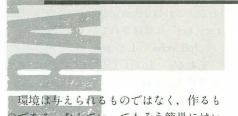
コマンドシェルを使っていて，同じものを何度も入力するのと，[CTRL]うんちゃらというコマンドを覚えるのと，どっちが楽か。それは人それぞれだろうな。楽だと思った機能だけ使えばいいのだ。

これで使いやすくなったといっても，エディタのようにカーソルが上下に動くわけではない，いうなればまだ1次元の世界なのだ（ヒストリがないときは，自由に動ける方向がひとつもない，いわば0次元だったのか）。この時代にDOSのオペレーションは1次元だというのは，変だと思わないか。こんな対症療法ではなく，オペレーティング環境の刷新がされないかな，と思う今日このごろである。マックにしろなんていったりはしないけれどね。

自分だけのおいしいコマンドライン

エイリアス主義のすすめ

Ogikubo Kei
荻窪 圭

Human68k ver.2.0にはたくさんの拡張機能があります。そのなかで，自分だけのコマンドを簡単に作れるエイリアス機能を重点的に見ていきましょう。単に名前を変えるだけでなく，マルチ機能と併用すれば新しい世界が開けてきます。

環境は与えられるものではなく，作るものである，なんていってもそう簡単にはいかない。たまたまcommand.xが気に入らないからといって，ユーザーが自分でシェルを作れるわけではない。しかし，ある程度与えられたもので，できるだけ自分のお気に入りのものへと近づけていくというのは大切な姿勢だ。

エイリアス入門

環境とひと口にいってもピンからキリまであって，画面表示からマウスの操作，コマンド体系，デバイス管理まで，ありとあらゆるものが対象にできるが，システム本来の思想を失わない範囲で自分のスタイルにあった環境を作ってしまいたい。そこで，よいシステムのひとつに，各自勝手に設定できるものが適度にあること，という条件を入れてもよいだろう。

たとえば，X68000付属のワープロで，毎回“自動逐次変換”で立ち上がるのは一括変換を使う私には困ったものである。これが“一括変換”で立ち上がるように設定できたらと思う人は多いだろう。また，かな漢字変換で使用するキー(ASKもVer2では変更可能となった）やら，画面の色（なぜかX68000のソフトではたいてい自由に変えられる）やら，マウスのダブルクリックの間隔(MacのOSやZ'sSTAFF PRO-68Kでは変えられる)やらといろいろある。

そしてHuman68kのバージョン2には，ユーザーレベルでコマンドシェル環境をよくする機能が付いた。ヒストリドライバ(history.x)を組み込むことで利用できるエイリアス機能である。

エイリアスというのはALIASと書く。ALIASとエイリアスがピンとこない人は，ALIENをエイリアンと読むことを思い出してほしい。アリエスと読んじゃう人はローマ字と間違えているのである。アライアスと読んじゃう人は，ダライアスのやりすぎである。

エイリアスというのは別名とか偽名という意味である。具体的には，コマンドの並びに何か名前をつけて，それで呼んでしまおう，ということである。エイリアスが優秀なのは，デバイスドライバとしてhistory.xを登録しておくと，そこで指定されたエイリアスのファイルをメモリバッファ上に読み込んでしまい，常用できることにある。もちろん，コマンドラインから指定することもできるし，メモリ上にあるエイリアス定義をファイルにしまうこともできる。

エイリアス機能は[ESC]-Iで制御する。

名前　定義[ESC]-I

はエイリアスの登録，ただの[ESC]-Iはエイリアスの定義リスト，

名前[ESC]-I

は，その名前のエイリアスの削除である。

では，具体的に見ていこう。

別名をつける

まずは文字どおり“別名”を与えるためのエイリアスから。

files dir

よーするにこれは，dirをfilesという名で定義するものである。これによって，filesと入力しても，ディレクトリが見られる。どーしてもBASICと同じコマンドが使いたいという人ならするかもしれない（私は逆に，BASICでdirと打ってしまってイラだつ)。

同様のパターンで，

list type

というのもある。さらに，UNIXユーザーなら，

ls dir

なんてするかもしれない。

よく使うコマンドは，あっさりとすませたい。そんな貴方なら，こうするかもしれない。

cp copy

copyと4回もキーを打つのが面倒なので，cpに減らしたわけだ。さらには，こんなこともするかもしれない。

sw switch

そりゃ，switchと打つよりswのほうが簡単だもんね。究極の手として，よく使うコマンドをすべてアルファベット1文字で表現してしまうというのもオタッキーでいいかもしれない。cはcopyで，dはdir，mはmove，wはwpという感じだ。delもdだとか，chdirもcといった問題もあるが，それは適当にほかの文字にふっちゃえ。

とまあこのへんは趣味の問題だが，“コマンドのオプション付き実行は面倒臭い！”という人は多いだろう。そんな人にもお勧めはエイリアスである。たとえば，

dir dir /p

である。コマンドと同じ名前で定義したときはどうなるか。エイリアスのほうが優先されるのである。優先順位は高いほうからエイリアス，内部コマンド，外部コマンド(拡張子.rのほうが.xより先)，バッチファイルなのだ。エイリアスは最初にチェックされるのである。

上の例は私も使っている。普通にdirを打てば，1ページごとに“どれかキーを押してください”といってスクロールが止まるので，ファイルが多いときに[CTRL]+Sに指をかけて目を凝らす必要もない。こういった使い方は有効だろう。普通にdirしたいときにはコマンドを入力した段階で[ESC]-*を打ってから実行すればいい。エイリアスの内容を無視して，元々のdirが実行できる。

このパターンでもっと便利なのが，

ed ed /e /l

である。エディタはやっぱり改行マークとEOFマークを表示して使いたいから，それをデフォルトにしてしまうのだ。

さらに，オプションがいろいろあるcopyallなんかも狙い目だろう。typeと打つとdelを実行し，delするとdumpするようなシュールな世界も作れるがこんな危ないことは試さないように（これでは別名じゃなくて偽名だな）。

新しいコマンドを作る-その1

これは楽しい作業である。

```
root cd ¥
```

私のHuman68kではrootと入力すると，どこにいてもルートディレクトリにカレントが移ることになっている。cd ¥と打っても手間は変わらないのだが，rootと打つとrootに戻るところが自然言語チックでいいんではないかと思う。

同様なコンセプトとして（きっと誰もがやってる気がするけど），

```
talk copy %1.pcm pcm
play copy %1.opm opm
```

てのも気持ちがいい。エイリアスでは，バッチファイル同様パラメータを渡すことができるのでこんな芸当が光るのである。ただ，“talk ファイル名”だけでよいのだ。「うちのパソコンはtalkコマンドでしゃべるんだぜ」とやってもよい。許す。

さて，もっと怠慢に生きたい私としては，よく変更するファイルに対しては専用のエディタコマンドを用意している。

```
edauto ed ¥autoexec.bat
edcon ed config.sys
```

の2つだ。この2つはよくいじるから，専用のコマンドにしてしまった。

さて，問題である。このedは，前述のエイリアス定義したed (ed /E /L)で立ち上がるのか，それとも，そのままed.xを読みにいくのか。これはconfig.sysによる。history.xの立ち上げ時専用オプションに，エイリアスの中でエイリアスを使えるか否か，というのがあって，どちらでも選べるのだ。

/arオプションを付けてあると，エイリアス定義の中でも，エイリアスしたコマンドを使える。/arを付けないと，エイリアスの中でエイリアスしたコマンドは使えない。つまり，

```
edc ed /e /l
edauto edc ¥autoexec.bat
```

の2つが登録されているとき，/arしてないと，edautoは“コマンドまたはファイル名が見つかりません”エラーになるが，/arしてあれば，うまくいくわけだ。

さて，味をしめた私は，こんなのも登録してしまった。

```
dbat dir *.bat
dx dir *.x
ddir dir *.
```

などである。ddirというのは，たいていのファイルには拡張子が付いているが，ディレクトリ名には普通付けないという習性を利用したものだ。

```
denv dir %%1%
```

もある。これは，環境変数%1にsetしたパスのディレクトリを見るものだ。

C言語を使うときには，

```
cy cc /Y %1.c
ce cc /E %1.c
```

なども手軽。

新しいコマンドを作る-その2

エイリアスはバッチファイルと違って，複数行に渡るコマンドを使うことはできない（ifなどのコマンドを使うことは可能だが）。しかし，｜｜でマルチ処理機能（BASICでいうマルチステートメント）したり，｜でパイプラインしたり，＞や＜でリダイレクトしたりはできるから，単純なバッチファイルを作るくらいなら，ずっとエイリアスのほうがいい。

いま重宝しているのは，

```
mcd md %1 || cd %1
```

である。ディスクからファイルをデベベっとコピーしたいとき，コピー先にディレクトリを作ってそこに放り込むことが多いので（ハードディスクを使っていると特に多い），それを楽しようと思って作ったのだ。受け側で“mcd ディレクトリ名”を実行し，あとはコピー一発である。

同じ発想で，

```
deld del %1¥*.* || rd %1
```

も重宝している。ディレクトリ削除のとき，便利だ。さらに深くサブディレクトリがある場合には使えないけれど，困ったことはとりあえずない。

ついでに，swapコマンドもよく使う。

```
swap ren %1 temp.$$$ ||
ren %2 %1 || ren temp.$$$ %2
```

である。何をするかというと，ファイル%1とファイル%2の名前を中間ファイルを介して入れ替えるのである。

さらに高度な技としては，エイリアスにコントロールコードを登録するというのがある。これはHuman68kのサンプル“history.his”に登録されている

```
lpff echo ^L > prn
```

である。^Lは[CTRL]+Lであり，プリンタに送ると改ページしてくれるのだが，こういったコントロールコードを入力することもできるのだ。そのときは，[ESC]-[CTRL]+n (nは任意の文字) 技を使う。すると，そのコントロールコードは実行されずに，画面に表示される。

また，Human68kのサンプルには，

```
apath path %path%;%1
```

というパスに新しいパスを追加するというおいしいのも付いている。このpath %path%;追加するパス，というのはhistory.Xなしでも使用できる。といっても，コマンドラインからでは使えない技なのだが，バッチファイル内とエイリアスでは有効なのだ。

これだけ簡単に自分のコマンドが作れるのもエイリアスのおかげである。このほか，アプリケーションの立ち上げにも，バッチファイルを作るより簡単でよろしい。

ちなみに，ALIASというのはUNIXのシェルが持っている機能である。(UNIXでは小文字のalias) ここでもUNIXの環境が手本となっていたりするのである。

＊　＊　＊

コマンドシェルを使いながら，「あ，これはさっきも打ったな」とか「これ，長くて打つのが面倒だな」と思ったとき，気ままに登録していくのがいい。そして，登録したエイリアスを“history.his”へ登録するのも，気が遠くなるほど多いhistory.xのオプションを覚えるのも面倒だから，エイリアスで，

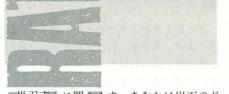
```
saval history /waa¥his¥history.his
```

を使う。そんなこんなで私のエイリアス定義は何十個(そういえば，学生時代UNIXを使っていたころも，やたらalias定義してた気がする。UNIXのコマンドがなかなか覚えられなかったから）にもなってしまった。もの覚えが悪い私がいま一番よく使うのはただの“[ESC]-I”(エイリアス定義一覧)だったりして，これもまた情けない話ではある。

コマンド操作からシステムコールの使い方まで

OS-9プログラミング教習所

Komura Satoshi
古村　聡

OS-9/X68000では用意された環境の範囲内で作業することはあまり快適ではありません。PDSなどを揃えられる状況でない限り，あとは自分で改善していくしかありません。地道にアセンブラの使い方などから見ていきましょう。

単刀直入に聞きます。あなたは以下のどれに該当しますか？

1) OS-9もMW-Cも買ったのである。もはや完璧に使いこなしていてなんの不自由も感じてないのだ。
2) OS-9なんか持ってないよぉ。
3) OS-9を買ったけどよくわかんない。まだCも買ってない。

1)，2)の方にはなーもいうことがありまへん。さっさと読み飛ばしていただいて結構です。

で，3）の方なんですが，実をいうと3）の方を対象に書きますのでとくとご覧あれ。少なくともつまずいたときの資料にぐらいはなるつもりです。

さて，私もいろいろと考えたのですがせっかくOS-9を買ったのにつまずいてしまった理由として次のことが考えられます。

"OS-9のコマンドの使い方がよくわからない"
"OS-9に標準でアセンブラがついていることを知らない"
"アセンブラがあってもユーザーコールの使い方がわからない"

まず，OS-9を買ってくると厚さ5cmのユーザーズマニュアルがついてくるのですがこいつがわかりにくい。私のようにOS-9を少しでも使ったことのある人間にはなんとか使い方がわかるのですが，きっと初めてOS-9を触る人にはどこから読んでよいものやらさっぱりでしょう。そこで，OS-9のコマンドを使って実際にOS-9のマニュアルの読み方がわかるぐらいになるまで，とりあえずコマンド入門をやってみたいと思います。

次にアセンブラについてなのですが，はっきりいってOS-9には一応ラインエディタとアセンブラがついてきているのですがこれほど使えないものもない。なにしろアセンブラについてはマニュアルにはひと言も書いてないし（まぁ，福袋だと思えば当然なのかもしれないが），HumanでのDOSコールに当たるユーザーコールについては売っている資料さえほとんどない。実は私もFM用に別売りされている資料を訳して使わざるをえなかったのです。

で，アセンブラの使い方，アセンブラプログラムの約束事，ユーザーコール一覧表という具合でやっていこうと思います。ページ数の都合でかなり駆け足になりますが，29,800円もかけたOS-9のためです。がんばってついてきてください。

実習の前に必要なこと

OS-9というOSはマルチタスクだマルチユーザーだとウリというか特徴となるような機能が非常に多いのですが，それが逆にOSのとっつきにくさ（つまりどこからいじっていいかわからない）となっているような気がします。

で，今回は「とりあえずHumanを知っていればなんとなく使える」という部分を中心に説明をやっていきたいと思います。HumanにしろOS-9にしろ「ディスクの管理」「プログラムを組む人のための機能を備える」というOS（DOS）としての目的に差はないわけですし，とりあえず触って機能の一部でも使い始めればそのほかの部分は自然と興味が出てきて自分でいろいろ調べて使ってくれるだろうからです。

さて，OS-9にはコンソールシステムが3つあります。AVライダーはほっといて，マルチスクリーン，パーソナルウィンドウのどちらを使ってもこの教習ではほとんどかわりませんが一応ここでは基本的にパーソナルウィンドウを使っているものとして話を進めていきます。

これからいろいろとディスクを書き換えますので一応，フロッピーディスクを使っている人は，

```
$ format /d0
$ make_d0_wnd
```

をしてバックアップをとっておいてください。

では，まずディレクトリを見ていこうかと思います。OS-9もHumanと同じく階層ディレクトリを採用しています。ディレクトリの書き表し方は，Humanが"¥"と"/"の両方を許していたのに対し，"/"のみとUNIXに近いものになっています。

たとえば，Humanでいう，

a:¥user¥game.s

というディレクトリは，

/d0/user/game.a

という具合に"/"を使って表します。

またドライブ名もA，B，C……ではなく，ハードディスクが/H0，フロッピーは/D0，/D1となっていてどのディスクから立ちあげてもハードディスクのドライブ名とフロッピーのドライブ名が入れ替わるようなこともありません。

また，

../ディレクトリ

で，ひとつさかのぼってみたディレクトリ

./ディレクトリ

で，現在位置から見たディレクトリを意味します。*がワイルドカード。拡張子はありませんから"."もファイルネームに使えます。

ディレクトリを見るコマンドはHumanと同じくdirです。

さて，前にも話したことがあるような気がしますが，OS-9のシェルにはほとんど内部コマンドというものがなく大半のコマンドがディスク上のあるディレクトリに入っています。

ディスクを立ちあげてすぐにdirを取ってみればわかりますが，CMDSというディレクトリにほとんどのコマンドが入っています。

dir　/d0/cmds

でももちろんコマンドは見られますがここ

はひとつ，デフォルトのディレクトリをCMDSに変えてしまいましょう。この暗黙に了解されるディレクトリをカレント作業ディレクトリと呼んで，

chd ディレクトリ

で変えることができます（HumanやUNIXのcdと同じですね）。

chd CMDS
dir

と入力してみてください。これがコマンドたちです。

使い方を見てみたいなら，

HELP コマンド名

で簡単な使い方を見ることができます。わからないコマンドがあったらこれで調べてみてください。

さて，このCMDSにあるコマンドがコマンドラインから入力して使うことができるのは，このCMDSが実行ディレクトリに暗黙のうちに指定されているからです。実行ディレクトリというのはその名のとおり実行するコマンドが入っているディレクトリで，これは変更することができます。

あまりやってほしくはないのですが，chxというコマンドがそうです。

chx ディレクトリ名

あまりやってほしくないというのは，いきなりchxをやってしまったらdirも使えなくなるからなのです（dirはCMDSにあったでしょ）。これをやってしまうと，使えるのはchx自身とchdくらいになってしまいます（chxをCMDSに戻せりゃ問題ないけど）。気をつけてほしいコマンドです。

HumanやUNIXではmkdirでディレクトリを作ったようにOS-9では，

makdir

でディレクトリを作ります。

Humanでのtype命令はOS-9では，

list ファイル名

にあたります。type命令で文字がずらーっと出てきたのをCTRL＋Sで止めたようにOS-9でもCTRL＋Sで一時停止できます。再開はCTRL＋Qです。また，

CTRL＋W

でも止められます。こちらはそれ以外のキーさえ押せば再開できます。

Humanのようにファイルへの入出力をほかのキーボードなどへの変更（リダイレクション）もあります。その際ファイルの終わりとしてCTRL＋ZをHumanで押したようにOS-9ではCTRL＋［で終了します。

HumanのF3キーで直前に押したキー内容が戻ってきましたね。あれはOS-9ではCTRL＋Aです。

そうそう，さっき出てきたリダイレクションはHumanと同じで<，>，パイプラインは！となっています（ただし，リダイレクションやパイプ記号の前後にスペースを入れてはいけない）。

ブレイクなんかもききます。

003：キーボードインタラプトがかかりました

といって止まるはずです。

それから，OS-9にもcopyがあります。オプションの－wなんかはHumanより使えます（自分でhelp copyしてみてね）。

ファイルを見る，削る……といったコマンドはどのOSも比較的似たようなものなのでとっつきやすいのではないでしょうか？

エディタについて

さて，OS-9のスタートアップファイル（Humanならautoexec.batですね）を書き換えて“自分のOS”っぽくしてみたいなー，などと考え始めたら今度はエディタが必要になってきます。Humanのedほど立派なものじゃないけどOS-9にも一応ついてます。edtってやつがそうです。

edt ファイル名

で起動します。こいつはラインエディタなので行単位にいろいろやってやらなくちゃいけない。ま，とりあえず，よく使うものだけは書いておくから自分で調べて使ってみてedtに慣れていってください。マニュアルには221～223ページに載っています。

L＊

とりあえず最初から最後まで見てみる。行数の後ろにその行の内容がくる。＊がついているのがいま編集の対象になる行。

行数

その行にポインタが移る。

＋（－）数字

その数字分ポインタを下（上）の行へ。

S/文字列/

文字列を探す。

C/文字列1/文字列2/

文字列を置き換える。

D数字

その数字の指す行を消す。

プログラム編

とりあえずディスクやディレクトリの操作とエディタが使えればプログラムを組める可能性があるわけです（マシン語を知っていればの話だけどね）。プログラムが組めれば，自分でコマンドを作って徐々に環境整備することもできます。いずれはエディタやシェルまで……とオペレーティングの基礎固めはプログラミングからです。

OS-9／X68000は標準ではなにも高級言語がついてきません（ほかの機種ではCがついてくる。もっともそれらの機種のOS-9は10万円以上するわけだけど）。で，ここではOS-9についてくるアセンブラを使ってのプログラミングについて説明しようと思います。

OS-9のマニュアルにはアセンブラの使い方はおろかアセンブラがついてくることさえ説明されてはいません。

X68000のOS-9のシステムディスクには，AV DATAのディスクというのがあります。もうすでにご存じの方も多いかとは思いますがこのディスクのディレクトリ.giftboxにはマクロアセンブラr68，リンカl68，ライブラリusr.l，そしてoskdef.dが入っています。.giftboxは隠れディレクトリになっていますのでそれらのツールはほかのディレクトリにまとめてコピーしておくのがよいでしょう。私は/h0/userというディレクトリを作っていろいろなソースリストとともに入れてあります。

ソースリストを書く

では，さっそくプログラムのソースリストから書いていきたいと思います。まず，なんらかのエディタを使うことになりますがどの方法でも構わないでしょう。

1）　スクリーンエディタを入手する

これがあればいちばん手っ取り早いのですが……はっきりいってスクリーンエディタを持っている人はCを買ったか，自分で作ったかしている人だと思います。高級言語があるのなら（まして，それがアセンブラの代わりになるようなものであれば）そっちを使ったほうがいいと私は思うのです

が……。なんにしても，スクリーンエディタを持っているのであればそちらのマニュアルを見てプログラムしてください。

2)　ラインエディタを使う

OS-9に標準でついてきたラインエディタedtを使う方法です。エディタの使い方は前のコマンド編で書きましたのでそちらを見ながらプログラムしてください。スクリーンエディタ（Humanのedなど）に慣れてしまった人にとっては多少使いにくい部分もあるかもしれませんが慣れてしまえばこっちのもんです。

3)　hufileを使う

私も，OS-9のエディタがあまり速くないため使うことがあるのですが，いったんHuman上のスクリーンエディタを使ってプログラムを作りhufileでOS-9上に持っていくという手があります。この方法はアセンブラのように行数の多くなるタイプのものを打ち込むときには有利な手でしょう。今回の「数あて」のほうはその方法を使いました。Human上のエディタである程度書いておいて，あとでOS-9上にhufileコマンドで持っていきr68でアセンブルしたあとデバッグはOS-9上のエディタで書き換える，といった具合です。

まず，Humanで書いたソースプログラムをフロッピー上に落とします。次にOS-9を立ちあげて，

XOS9>hufile　/d0

とします。

で，たとえばソースリストの名前がgame.aであれば，

hufile>get　－r game.a

としてやれば，OS-9の作業ディレクトリのほうにgame.aが入っているはずです。

逆にOS-9→Humanのときは，

hufile>put　－r game.a

です。

hufileのサブコマンドにはHumanの内部コマンドでよく使うものはだいたい入っています。dirも見ることができますし，もちろんtypeもできます。

OS-9上でのプログラミング上の注意

OS-9の大きな特徴にマルチタスク・マルチユーザーであるということがあります。このマルチタスク・マルチユーザーであるがためにOS-9ではシングルタスク・シングルユーザーであるHumanとは違って，ある違った状況が起こりえます。つまり違ったユーザーが（または違ったタスクで）同じモジュールを使うことがあるのです。たとえばメッセージを出力するモジュールがあったとしましょう。ユーザー1が“こんにちは”とユーザー2が“Hello”と同時に表示したがったとしましょう。このような場合（厳密にいうとちょっとニュアンスが違うような気もしますが）OS-9では同時に同じモジュールが文字出力を受け持つのです。

このような事態を解決するためにOS-9でのプログラミングはできる限りリエントラントにしなければならない，という制約がつきます（実際には共有使用を禁止することもできる。つまり，絶対リエントラントでなくてはならないというわけではないが，できる限りリエントラントにする）。

たとえば，先のメッセージ出力モジュールの場合ならワークエリアを自分自身の中に持たなければよいのです。

モジュール側でワークエリアを持っていたとすると……。まず，ある処理で表示するメッセージ1をワークに格納し最初の文字を表示しました。次の処理でも同様にメッセージ2を表示しようとすると……メッセージ2をワークエリアに格納するときにメッセージ1を破壊してしまいます。しかし，モジュールが呼び出されるたびに別の変数エリアを確保していればそうはなりません。ユーザー1から呼び出されメッセージを表示しながらユーザー2の表示しようとするメッセージを同時（厳密には時分割して，メッセージ1を何文字か出してから2を数文字出して，1を出してとやっているんだけど）に出してくれます。

まさに，こういう理由で，OS-9のプログラムでは各種宣言部，変数領域，プログラム領域と分けて書かなければなりません。絶対にプログラム領域内でメモリの内容を書き換えてはならないのです。また，絶対にOSのユーザーコールを介さないでI/Oのアクセスをしてはなりません。I/Oをある一定の操作を行うグローバルワークエリアのようなものと思えば当然でしょう。OS-9自体が，「プログラムモジュールはプログラム領域は書き換えない。I/Oもいじらない」という前提のもとに作られているのでそのようなお行儀の悪いプログラムを走らせてしまうと動作の保証ができないのです。これは同じくマルチタスクのOS/2などでもそうなのだそうです。

サンプルプログラムを見ればわかるかと思いますが，変数領域はvsectという疑似命令で定義します。

vsect　～　ends

この2つに囲まれた部分が変数領域です。この中の変数は書き換えて構いません。

いま，使っているリロケータブルマクロアセンブラには変数関係の疑似命令として次のようなものがあります。

● **dc（.b.w.l）　データ値，データ値…**

dcは初期値を設定するデータで文字列の設定などに使います。

dc.b　“文字列”,$0d

などとすれば文字列データも入ります。「数あて」のリスト2を参照してください。

● **dz（.b.w.l）　サイズ**

これは初期値が0の場合のワークエリアの確保です。

● **ds（.b.w.l）　サイズ**

これは初期値が設定されないワークです。

この3つのうちdcはプログラム領域に書いてデータを参照することもできます。リスト1のパラメータを表示するだけのサンプルでもそのように使用しています。が，先ほどからしつこくいっているように，絶対にプログラム領域を書き換えるという暴挙には出ないでください。

さて，先ほどもいったようにOS-9では自分のほかにも誰かが同じコマンドを使っているかもしれず，そのため変数領域もどこにあるかわかりません。どうやってプログラムはワークエリアをアクセスするので

リスト1　引数の受け渡し

```
===============  yasa.a  ===============
    1:     use "oskdefs"
    2:     psect yasa,(Prgrm<<8)+Objct,
  (ReEnt<<8)+Rev,Edition,Stack,main
    3: Rev  equ 1
    4: Edition equu 1
    5: Stack equ 1
    6: main:
    7:     move.l a5,a0
    8:     move.l d5,d1
    9:     move.l #1,d0
   10:     os9 I$WritLn
   11:      cmp.b #1,d1
   12:      beq   Err
   13:     clr.w d1
   14:     os9 F$Exit
   15: Err
   16:     lea.l mes(pcr),a0
   17:     move.l #mesend-mes,d1
   18:     move.l #1,d0
   19:     os9 I$WritLn
   20:     clr.w d1
   21:     os9 F$Exit
   22: mes dc.b "パラメータがありません",$0D
   23: mesend
   24:      ends
```

しょう。ここに68系MPUを使っている強みとr68のr68たる部分があります。68系のMPUの命令はアドレッシングモードが非常に豊富です。そこでOS-9/68000ではアドレスレジスタ（a6）を介してアクセスします。OS-9の変数領域は(a6)の指す番地から前後あわせて64Kバイトに広がっています。68000のマシン語のアドレッシングにはインダイレクト・ウィズ・ディスプレースメントというモードがあってd16(An)と書きますが，これはd16の部分に16ビットのオフセットを書き込んでおけば，

データ← （Anの指す番地＋オフセット）

というデータを指してくれますのでOSの教えてくれるデータ領域の中心線A6を使えばきっちりほかとはぶつからないでアクセスできるのです（そのため，Humanのプログラムに比べると多少遅く，そして大きくなってしまう）。

そしてA6から目指すワークエリアまでのオフセットはラベルさえ書いておけばr68がちゃんと計算してくれますのでプログラムを書くときには，たとえばLEA命令であれば，

```
lea.l   work (a6),a0
```

と“ラベル（a6）”のナントカのひとつ憶えでOKです。

プログラムセクション内の参照は“ラベル（pcr）”のPCリラティブモードで参照します。PCリラティブとは現在のプログラムカウンタをアドレスレジスタの代わりに用いるもので相対的に番地を指定してデータを参照するためプログラムがメモリ上のどこに位置していてもデータを参照できるという利点があります。これとBcc，BRA命令を併用すれば特別なことをしなくてもプログラムをリロケートできるという利点があります（Humanのr形式と同じです）。

プログラムセクション

さて，サンプルを見るとどちらにも変数領域の前にごちゃごちゃとなにか書いてあります。

```
use "oskdefs"
psect yasa, (Prgrm<<8)+Objct,
(ReEnt<<8)+Rev, Edition, Stack,
main Rev equ 1
Edition equ 1
Stack equ 1
main:
 ∫
ends
```

これらもr68（つまりOS-9）でプログラムを書く際には必須のことがらですので解説します。

まず，use命令ですがこれはほかの言語などで見たことのある方も多いことでしょう。そうです，turboPascalなどでライブラリのインクルードに使うあれとまったく同じです。先ほど，.giftboxディレクトリから取り出したoskdefsを参照するための宣言です。

このなかにこのあとのpsectで指定する諸々のラベルの内容が入っています。

psect命令はプログラムセクションの始まりを定義する命令で普通アセンブラでプログラムを組むときには，

psect ～ ends

となっています。

パラメータはまず左からセクション名。適当な名前をつけてください（カッコ内は実際の数値）。

次にモジュールタイプと使用言語。

［モジュールタイプ］

	(0) システムコールのワイルドカード
Prgrm	(1) プログラムモジュール
Sbrtn	(2) サブルーチンモジュール
Multi	(3) マルチモジュール
Data	(4) データモジュール
	(5-10) リザーブ
TrapLib	(11) ユーザートラップライブラリ
System	(12) システムモジュール
Flmgr	(13) ファイルマネージャ
Drivr	(14) デバイスドライバ
Devic	(15) デバイスディスクリプタ
	(16-255) ユーザー定義領域

［言語］

	(0) システムワイルドカード
Objct	(1) 68000マシン語
ICode	(2) MW-BASIC（Iコード）
PCode	(3) MW-Pascal（Pコード）
CCode	(4) C言語（Iコード）
CblCode	(5) COBOL（Iコード）
FrtnCode	(6) FORTRAN（Iコード）
	(7-15) システムリザーブ
	(16-255) ユーザー定義領域

これらを上位，下位それぞれ8ビットに割り振って宣言します。今回はプログラムでマシン語ですから，

(Prgrm<<8)+Objct

です。

次が属性で，このプログラムがリエントラントであることを指しています。

(ReEnt<<8)+Rev, Edition

リエントラントであることを指している場所の下8ビットとその次の部分は自分で定義していることからもわかると思いますがこれはこのプログラムが第何版であるかを示しているものです。最初のバージョンなので両方とも1にしましたが特に気にする必要はありません。

最後の2つもなんとなくわかるでしょう。stackはその名のとおりスタックサイズです。最後のmainはプログラムのエントリポイント。そのラベルの示す位置からプログラムがスタートします。したがって，Pascalのようにいちばん最後のプロシージャから始まるようなプログラムももちろん作れます。

起動時のレジスタ内容

いまはOS-9のコマンドラインから使うプログラムをサンプルを使って説明しているのですが，OSからユーザープログラムに渡されるパラメータがありますのでそれについて述べておきましょう。

これらのパラメータはシステムコールのF$Forkを使って子プロセスの起動をしたときも同じです。パラメータはレジスタを使って渡されます。

d0.w	プロセスID
d1.l	グループ・ユーザーID
d2.w	優先順位
d3.w	I/Oパスの数
d5.l	パラメータのバイト数
d6.l	すべてのデータ領域の大きさ
(a1)	データ領域の再上位
(a3)	モジュールの先頭
(a5)	パラメータの先頭
(a6)	データ領域の中心
(a7)	a5のコピー

いちばんよく使うのが（a6），それにd5，（a5）というところでしょうか。サンプル1“パラメータを表示するだけのプログラム”を参考にしてください。

システムコール

さて，プログラムの本体，つまりMOVEでデータを転送してADDで足し算をしてという部分は別にHuman上でのプログラミングとなんら変わるところはありません。ただ，リエントラント・リロケータブルをこころがける，つまりワークエリアの書き換えなどは“WORK (a6)”，プログラム本体内の定数値の参照には“table (pcr)”と書くようにすればよいだけです。残りの問題はひとつ。どうやってI/Oにアクセスするかだけです。

これはOS-9のシステムコールを使うしかありません。これでOPENして，READ，WRITEしてCLOSEするのです。OS-9のシステムコールは次の3つがあります。

1)　ユーザーモードリクエスト
2)　I/Oリクエスト
3)　システムモードリクエスト

このうちユーザーモードのプログラムで使用可能なのは1)と2)です。

先ほどのI/Oをアクセスするようなときに使うのは2)のI/OリクエストでOPEN，CLOSE，WRITE，READからmakdir，deleteといったコマンドでお馴染みのものまであります。

で，1)のユーザーモードリクエストなのですがカーネルで処理しているOSの機能をユーザープログラムにいってみれば，分け与えてくれるものです。これには子プロセスの実行Fork，時刻を知るTime，プログラムの終了Exitなどがあります。

1)のものにはシステムコールの名前の頭にF＄が，2)のものにはI＄という名前がついています。

システムコールの名前と機能については最後に一覧表をつけておきましたのでそちらを参考にしてください。なお，ほかの関連知識やオプションが必要なシステムコールには書いてないものがあります。また，この表はマイクロウェアが出しているテクニカルマニュアルやその他の資料を基に私が編集したものなので若干おかしな部分があるかもしれませんが，そのときはOh!X編集室までハガキでご連絡ください。

ところで，これらのシステムコールの呼び出し方なのですが，これにはr68の内蔵のマクロを使って呼び出します。

まず，表を参考にしてレジスタに各種の値を設定してください。そのあと，

　os 9　I$Write

などのように“os9”という疑似命令を使って呼び出します。

```
例)  lea.l    mes5 (a6),a0
     move.l   #mes5e−mes5,d1
     move.l   #1,d0
     os9      I$WritLn
```

アセンブル，リンク

もうここまでわかればこっちのもの。あとはプログラムを書いてアセンブルするだけです。アセンブラの名前はもう何度も出てきましたがr68といいます。で，このr68は，

　r68　ファイル名

だけではオブジェクトファイルを作ってくれません。

　r68　ソース名　−O＝オブジェクトファイル名

で作ります。

リンカの168も，

　168　オブジェクト名　−l＝ライブラリファイル名　−o＝実行ファイル名

で，実行ディレクトリ（ふつうはCMDS）に実行プログラムを入れてくれます。

たとえばリスト2の数あてゲームの場合ソースの名前はgame.aですから，

　XOS9>r68 game.a　−O＝game.r
　XOS9>l68 game.r　−l＝usr.l -o＝game

とすれば，実行ディレクトリcmdsにコマンドgameができます。そうしたら，

　XOS9>game

とすれば数あてゲームができます。

r68もl68もオプションに−?をつけると説明が出てきますので（英語ですけどね。辞書と首っぴきでがんばって）読んでください。r68にはアセンブルリストの出し方が豊富にオプションでついてきます。

最後に

最後にこれらの話の参考になればと，非常に小さいものですが，サンプルプログラムを作っておきました。

ひとつめはただ単にパラメータを表示するだけのプログラム。

2つめは数あてゲーム。コンピュータがランダム（ただし同じ数字は2度使わない）な4桁の数字を持っていますからそれをあててください。こちらがなにかいえば，桁の合っている数字があれば1ヒット，桁は違うが使っているというのがあれば1エラーとして，○ヒット×エラーだよといってきます。

たとえばコンピュータの考えているのが1234でこちらが4130と入力すれば1ヒット2エラーと返ってきます（まったくあっていなければ「おおはずれー！」といってきます）。

ちょっと乱数の作り方が悪くて待たされますが（バックトラックにすりゃよかった……）気長に待ってください。

このサンプルが皆さんの参考になれば幸いです。

ところでusr.lにはまだT$LMULやT$DDivといった数値演算関係のコールも入っているようなのですが残念ながら手元にそれ関係の資料がないためどうやって使うのかわからないでいます。そのへんがわかったらまたOS-9関係のページにも書いていきたいと思っています。

ではそのときまで。

リスト2　game.a

```
================  GAME.A  ================
     1:     use "oskdefs"
     2:     psect game,(Prgrm<<8)+Objct,(ReEnt<<8)+Rev,Edition,
   Stack,main
     3: Rev  equ 1
     4: Edition equ 1
     5: Stack equ 1024
     6:
     7:         vsect
     8: *
     9: *変数領域
    10: *メッセージテーブル
    11: *
    12:
    13: mes0:
    14: hitc:    dc.b $30
    15:          dc.b "ヒット"
    16: mes1:
    17: errc:    dc.b $30
    18:          dc.b "エラー",$0D
    19: mes0e:
    20: mes2:    dc.b "大当りーっ！",$0D
    21: mes2e:
    22: mes3:    dc.b "おおハズれーっ！",$0D
    23: mes3e:
    24: mes4:    dc.b "4桁の数字をいれてください",$0D
```

```
 25: mes4e:
 26: mes5:    dc.b "＊＊＊数あて＊＊＊",$0D
 27: mes5e:
 28: mes6:
 29: kami:      dc.b $30
 30: kaisuu:    dc.b $31
 31:          dc.b "回目で当たりました。",$0D
 32:
 33: mes6e:
 34: mes7:    dc.b "答えは、 "
 35: n1:      dc.b $30
 36: n2:      dc.b $30
 37: n3:      dc.b $30
 38: n4:              dc.b $30
 39:          dc.b "  でした！",$0D
 40: mes7e:
 41: mes8:    dc.b "4桁の数字を入力してください",$0D
 42: mes8e:
 43: strbuf:
 44:          dc.l $20202020
 45:          dc.b $20
 46: sbe:
 47: ret:     dc.b $0d
 48:
 49:          ends
 50: *
 51: *プログラム領域
 52: *main program
 53: *
 54: main:
 55:          lea.l   mes5(a6),a0
 56:          move.l  #mes5e-mes5,d1
 57:          move.l  #1,d0
 58:          os9     I$WritLn
 59: mloop1:
 60:          bsr.s   ininum
 61: *        bsr.s   endmes            *デバッグ用メッセージ
 62: mloop2:  bsr.w   inpnum            *数字を入れさせる
 63:          bsr.w   chknum            *数字のチェック
 64:          cmp.b   #255,d0           *d0.b=255なら終り
 65:          beq.s   gamend
 66:          bsr.w   errprn            *
 67:          addq.b  #1,kaisuu(a6)
 68:          move.b  kaisuu(a6),d0
 69:          cmp.b   #$3a,d0
 70:          bne.s   jump1
 71:          move.b  #$30,kaisuu(a6)
 72:          addq.b  #1,kami(a6)
 73:
 74: jump1:   bra.s   mloop2
 75:
 76: gamend:  lea.l   mes2(a6),a0       *全部当たって最後のメッセージ
 77:          move.l  #mes2e-mes2,d1
 78:          move.l  #1,d0
 79:          os9     I$WritLn
 80:          lea.l   mes6(a6),a0
 81:          move.l  #mes6e-mes6,d1
 82:          move.l  #1,d0             *標準出力
 83:          os9     I$WritLn
 84:          bsr.s   endmes
 85:          clr.w   d1                *異常なしで
 86:          os9     F$Exit            *おわります！
 87:
 88: endmes:  lea.l   mes7(a6),a0
 89:          move.l  #mes7e-mes7,d1
 90:          move.l  #1,d0
 91:          os9     I$WritLn
 92:          rts
 93: *
 94: *ランダムな4桁の数を作る
 95: *
 96: ininum:  move.w  #1,d0
 97:          os9     F$Time            *乱数に時間を使う
 98:          move.l  d0,d1
 99:          and.l   #$0f,d1
100:          ror.l   d1,d0
101:                  and.l  #$ffff,d0
102:          move.l  #0,d1             *D1=保存用
103:
104:          bsr.s   div10
105:          move.b  d0,n4(a6)
106:          move.l  d1,d0
107:          bsr.s   div10
108:          move.b  d0,n3(a6)
109:          move.l  d1,d0
110:          bsr.s   div10
111:          move.b  d0,n2(a6)
112:          move.l  d1,d0
113:          bsr.s   div10
114:          move.b  d0,n1(a6)
115:
116: chkkas:  move.b  n1(a6),d0         *重なりチェック
117:          cmp.b   n2(a6),d0         *あ、手抜きだ。
118:          beq.s   ininum            *連載であんなことを
119:          cmp.b   n3(a6),d0         *書いたばっかりなのに
120:          beq.s   ininum
121:          cmp.b   n4(a6),d0
122:          beq.s   ininum
123:          move.b  n2(a6),d0
124:          cmp.b   n3(a6),d0
125:          beq.s   ininum
126:          cmp.b   n4(a6),d0
127:          beq.s   ininum
128:          move.b  n3(a6),d0
129:          cmp.b   n4(a6),d0
130:          beq.s   ininum
131:          rts
132:
133: div10:
134:          divu.w  #$0a,d0                *d0=d0 mod 10
135:          move.w  d0,d1                 *d1=d0 div 10
136:          lsr.l   #8,d0
137:          lsr.l   #8,d0
138:          add.b   #$30,d0
139:          rts
140:
141: *
142: *数字の入力
143: *
144:
145: inpnum:
146:          lea.l   mes8(a6),a0       *" 数字をいれてください"
147:          move.l  #mes8e-mes8,d1
148:          move.l  #1,d0
149:          os9     I$WritLn
150:          move.l  #1,d0             *標準入力
151:          move.l  #4,d1             *バッファは4bytes
152:          lea.l   strbuf(a6),a0
153:          os9     I$Read
154:          cmp.b   #4,d1
155:          bne.s   inpnum
156: latchk:
157:          move.b  #4,d1             *数字チェック
158:          lea.l   strbuf(a6),a0
159: latlop:  move.b  (a0)+,d0
160:          cmp.b   #$30,d0
161:          bcs.s   inpnum
162:          cmp.b   #$3a,d0
163:          bcc.s   inpnum
164:          subq    #1,d1
165:          bne.s   latlop
166:          lea.l   ret(a6),a0
167:          move.l  #1,d1
168:          move.l  #1,d0
169:          os9     I$WritLn
170:          rts
171:
172: errprn:  move.b  hitc(a6),d0
173:          bne.s   prnhit
174:          move.b  errc(a6),d0
175:          bne.s   prnhit
176:          lea.l   mes3(a6),a0       *" おおはずれー！"
177:          move.l  #mes3e-mes3,d1
178:          move.l  #1,d0
179:          os9     I$WritLn
180:          rts
181:
182: prnhit:  lea.l   mes0(a6),a0       *○ヒット○エラー"
183:          move.l  #mes0e-mes0,d1
184:          move.l  #1,d0
185:          os9     I$WritLn
186:          rts
187: chknum:
188:          move.b  #4,d0             *数字がおなじか？
189:          lea.l   n1(a6),a0
190:          lea.l   strbuf(a6),a1
191: chklop:  cmpm.b  (a0)+,(a1)+
192:          bne.s   chkerr
193:          subq    #1,d0
194:          bne.s   chklop
195:          move.l  #255,d0
196:          rts
197: chkerr:
198:          move.b  #$30,d0           *hitとerrorのカウント
199:          move.b  d0,errc(a6)       *d1=com's answer
200:          move.b  d0,hitc(a6)       *d2=man's answer
201:          move.b  #4,d1
202:          lea.l   n1(a6),a1
203: loop2:   move.l  #4,d2
204:          lea.l   strbuf(a6),a2
205: latlo2:  move.b  (a1),d5
206:          cmp.b   (a2)+,d5
207:          beq.s   inccnt
208: retb:    subq    #1,d2
209:          bne.s   latlo2
210:          lea.l   1(a1),a1
211:          subq    #1,d1
212:          bne.s   loop2
213:          move.l  #0,d0
214:          rts
215:
216: inccnt:
217:          cmp.b   d1,d2
218:          beq.s   inchit
219:          move.b  errc(a6),d3
220: inc2:    addq.b  #1,d3
221:          move.b  d3,errc(a6)
222:          bra.s   retb
223: inchit:  move.b  hitc(a6),d3
224:          addq.b  #1,d3
225:          move.b  d3,hitc(a6)
226:          bra.s   retb
227:
228:          ends
```

OS-9のシステムコール(システムモード除く)

この表は独自調査によるものです。内容についてシャープ，マイクロウェアには問い合わせないようにしてください。バグなどがありましたらOh!X編集室までお願いします。

ユーザーモードリクエスト

●F$AllBit

アロケーションマップのビットセットを行うためのシステムコールで，F$SchBitで得られた空き領域のビット0を1に反転します。RBFタイプのデバイスでのクラスタ管理に使用されます。

入力
d0.w =反転するビットの開始位置
d1.w =ビット数
(a0) =アロケーションマップの開始位置出力なし

●F$CRC

CRCの値を計算するファンクションコールです。最初に計算するときは，

move.l #$FFFFFFFF,d1

としてください。

入力
d0.l = 計算するバイト数
d1.l = CRCアキュムレータ
(a0) = 計算開始番地
出力
d1.l = CRCアキュムレータ

●F$Chain

プロセスディスクリプタをそのままにして実行モジュールを切り替えます。

入力
d0.l = モジュールの言語のタイプ
(0: ワイルドカード値)
(1: マシン語オブジェクト)
(2: MW-BASIC)
(3: Pコード (Pascal))
(4: Iコード (C))
(5: 〃 (COBOL))
(6: 〃 (FORTRAN))
(7-15: リザーブ)
(16-255: ユーザー定義領域)
d1.l = 追加データ領域
d2.l = パラメータバイト数
d3.w = コピーされるI/Oパス数
d4.w = 優先順位
(a0) = 実行モジュール名先頭
(a1) = パラメータ先頭
出力
(a0) =

●F$CmpNam

文字列のコンペア。比較先文字列は00で終わるASCIIコード列です(比較元はバイト数を指定する)。基本的にファイルネームのコンペアですのでワイルドカードの使用が可能で大文字小文字の区別はしません。

入力
d1.w = 比較元バイト数
(a0) = 比較元先頭番地
(a1) = 比較先先頭番地
出力
キャリフラグを上げ下げしますので，bcc(bcs)命令を使用してください。

●F$CpyMem

ほかのプロセスの使用しているメモリの内容をコピーします。

入力
d0.w = ほかのプロセスのプロセスID
d1.l = コピーバイト数
(a0) = コピー元アドレス
(a1) = コピー先アドレス

●F$DatMod

プロセス間のデータ通信を行うためのユーザーコールでそのためのデータモジュールを作成します。

入力
d0.l = データ領域の大きさ
d1.w = モジュールの属性・リビジョン
d2.w =
(a0) = モジュール名先頭
出力
d0.w = モジュールの言語タイプ
d1.w = モジュールの属性・リビジョン
(a0) = 名前の先頭
(a1) = データの先頭(オフセット)
(a2) = モジュールヘッダ先頭

●F$DelBit

F$AllBitでセットしたアロケーションマップのクリアをします。

入力
d0.w = クリアするビットの開始位置
d1.w = ビット数
(a0) = アロケーションマップの開始位置

●F$DExec

F$DForkで起動したプロセスの実行をします。

入力
d0.w = 実行子プロセスのID
d1.l = 実行命令数(0:連続)
d2.w = ブレイクポイント数
(a0) = ブレイクポイントテーブル先頭
出力
d0.l = 実行済み命令数
d1.l = 残り命令数
d2.w = 例外発生番地(オフセット)
d3.w =
d4.l =
d5.l =

●F$DExit

F$DExecで実行したモジュールの終了入力

入力
d0.w = 終了させる子プロセスのID

●F$DFork

トレースモードでの子プロセス起動を行います。主としてデバッグ用です。

入力
d0.w = モジュールの言語のタイプ
d1.l = 追加するデータ領域バイト数
d2.l = パラメータバイト数
d3.w = コピーするI/Oパス数
(a0) = プライマリモジュール名先頭
(a1) = パラメータ先頭
(a2) = 子プロセスのレジスタバッファ先頭
出力
d0.w = 子プロセスのID
(a0) = 更新後の名前先頭
(a1) = 子プロセスのレジスタバッファ先頭

●F$EXIT

プロセスを終了させます。

入力
d0.w = 呼び出し元へのステータスコード

●F$Fork

新しいモジュールの起動を行います。

入力
d0.w = 起動するモジュールの言語タイプ
d1.l = 追加データ領域バイト数
d2.l = パラメータ数
d3.w = コピーするI/Oパス数
d4.w = 優先順位
(a0) = プライマリモジュール名先頭
(a1) = パラメータ先頭
出力
d0.w = 子プロセスのID

●F$GPrDsc

指定したプロセスのプロセスディスクリプタを転送します。

入力
d0.w = 転送元ID
d1.w = 転送バイト数
(a0) = 転送先先頭番地

●F$ID

現在のID・優先順位を調べます。

出力
d0.w = プロセス
d1.w (h) = グループ
d1.w (l) = ユーザー
d2.w = 優先順位

●F$Icpt

キーボードの処理などに用いるインタラプト処理用ルーチン用のフックテーブルを設定します。

入力
(a0) = ルーチンの先頭
(a6) = ルーチン用のワーク領域先頭
出力
d1.w =0 システムアボート
1 プロセスの復帰
2 キーボードアボート
3 キーボードインタラプト
4 アラーム
それ以外はユーザーが再定義可能
(a6) = ワークの領域先頭

これらがインタラプトの起きたさいに処理ルーチンに渡されます。

●F$Julian

年月日→ユリウス日&0時からの秒数に変換します。

入力
d0.l = 時刻 (00hhmmss)
d1.l = 年月日 (yyyymmdd)
出力
d0.l = 0時からの秒数
d1.l = ユリウス日

●F$LINK

モジュールのリンクをします。

入力
d0.w = モジュールの言語タイプ
(a0) = モジュール先頭
出力
d1.w = モジュールのリビジョン
(a0) = 更新後の名前先頭

(a1) = モジュールの実行番地
(a2) = モジュールの先頭

●F$Load
ディスク上のモジュールをロードします。
入力
d0.b = アクセスモード
(a0) = パスリスト
出力
d0.w = モジュールの言語タイプ
d1.w = モジュールのリビジョン
(a0) = 更新後の名前先頭
(a1) = モジュールの実行番地
(a2) = モジュールの先頭

●F$Mem
データ領域の大きさに関するファンクションです。
入力
d0.l = 0 領域はいくらか?
<> 0 変更後の大きさ
出力
d0.l = 確保している領域の大きさ

●F$PErr
エラーを出力・表示します。
入力
d0.w =
d1.w = エラー番号
出力
エラーメッセージ

●F$RTE
インタラプト処理ルーチンを終了させます。

●F$SchBit
アロケーションマップから空き領域を探します。確保できない場合いちばん大きな空き領域を返します。
入力
d0.w =探し始めるビットの位置
d1.w =確保したいビット数
(a0) =アロケーションマップの位置
(a1) =アロケーションマップの最後+1出力
d0.w = 見つかったビット位置
d1.w = 確保できるビット数

●F$Send
ほかのプロセスへシグナルを送ります。
入力
d0.w =0 全プロセス
それ以外はそのID
d1.w =0 システムアボート
1 プロセスの復帰
2 キーボードアボート
3 キーボードインタラプト
4 アラーム
それ以外はユーザーが再定義可能

●F$Sleep
プロセスを一定時間休止状態にします。
入力
d0.w = タイムスライス数(最上位ビットがたっていたら1/256秒単位になります)

●F$SPrior
プロセスの優先順位の変更。
入力
d0.w = プロセスのID
d1.w = 新しい優先順位

●F$STime
日時の設定。
入力
d0.l = 時(00hhmmss)
d1.l = 日(yyyymmdd)

●F$Time
日時を得ます。
入力
d0.w =0 通常の暦
=1 ユリウス暦の日数
出力
通常
d0.l = 時刻 00000000hhhhhhhhmmmmmmmmssssssss
d1.l = 年月日yyyyyyyyyyyyyyyymmmmmmmmdddd
ユリウス暦
d0.l = 0時からの秒数
d1.l = 日数
共通
d2.w =曜日(0=日……6=土)

●F$UnLink
モジュールをアンリンクしモジュールディレクトリから削除します(アドレス指定)。
入力
(a0) = モジュールのアドレス

●F$UnLink
モジュールをアンリンクしモジュールディレクトリから削除します(名前指定)。
入力
d0.w = モジュールの言語タイプ
(a0) = 名前の先頭アドレス

●F$Wait
コプロセスの実行が終わるまでWAITします。
出力
d0.w = 終了した子プロセスID
d1.w = ステータスコード

I/Oリクエスト

●I$Attach
デバイスの登録と登録確認のためのコールです。
入力
d0.b = アクセスモード
(ビット1=1 書き込み
0=1 読み出し)
(a0) = デバイス名
出力
(a0) = デバイステーブルポインタ

●I$ChgDir
作業ディレクトリの変更を行います。アクセスモードの指定で実行ディレクトリの変更にもなります。
入力
d0.bのビット2=1 実行ディレクトリの変更
(a0) パスネーム
出力
(a0) = パスリスト

●I$Close
パスのクローズ。
入力
d0.w = パス番号

●I$Create
新しいファイルを作ります。
入力
d0.b = (ビット1=1 書き込み
0=1 読み出し
ただしupdateはともに1)
d1.b = (ビット6=1共有不可
5 一般実行可
4 〃 書き込み可
3 〃 読み込み可
2 オーナーのみ実行
1 〃 書き込み可
0 〃 読み込み可)
(a0) = パスリスト
出力
d0.w = パス番号
(a0) = 新しいパスリスト

●I$Delete
ファイルを削除します。
入力
d0.b = (ビット2 実行ディレクトリ
1 書き込み
0 読み込み)
(a0) = パスリスト
出力
(a0) = 新しいパスリスト

●I$Detach
デバイスの削除を行います。
入力
(a2) = デバイステーブル

●I$MakDir
新しいディレクトリを作ります。
入力
d0.b = (ビット7=1 ディレクトリ
2=1 実行ディレクトリ)
d1.b = (ビット6=1共有不可
5 一般実行可
4 〃 書き込み可
3 〃 読み込み可
2 オーナーのみ実行
1 〃 書き込み可
0 〃 読み込み可)
(a0) = パスリスト
出力
d0.w = パス番号
(a0) = 新しいパスリスト

●I$Open
パスをオープンします。
入力
d0.b = (ビット1=1 書き込み
0=1 読み出し
ただしupdateはともに1)
(a0) = パスリスト
出力
d0.w = パス番号
(a0) = 新しいパスリスト

●I$Read
データの読み出し(編集なし)。
入力
d0.w = パス番号
d1.w = 読み出したいバイト数
(a0) = バッファのアドレス
出力
d1.l = 読み出せたバイト数

●I$ReadLn
データの読み出し(編集あり)。
入力
d0.w = パス番号
d1.w = 読み出したいバイト数
(a0) = バッファのアドレス
出力
d1.l = 読み出せたバイト数

●I$Seek
ファイルポインタを設定します。
入力
d0.w = パス番号
d1.l = ファイルポインタの絶対位置

●I$Write
データ書き出し(編集なし)。
入力
d0.w = パス番号
d1.w = 書き込むバイト数
出力
d1.w = 書き込めたバイト数

●I$WriteLn
データ書き出し(編集あり)。
入力
d0.w = パス番号
d1.w = 書き込むバイト数
出力
d1.w = 書き込めたバイト数

「プロセス操作」という世界

Murata Toshiyuki 村田 敏幸

プロセス管理にメモリ管理。よく聞く言葉ですが，実際にはどうやっているのでしょう？　これらの操作は本来OSのみに許された機能です。それだけに奥が深く，知れば知るほど面白いものです。今回は，OSになったつもりで，これらの世界を垣間見てみましょう。

今回は“プロセス”関連の話題を取り上げる。プロセスという言葉自体は“チャイルド（子）プロセス”なんていう言い回しでお馴染みだろう。コンピュータ用語のプロセスは“一連の工程”ぐらいの意味で，あるひとまとまりの処理単位を指す言葉だ。Human68kでは基本的に1本のプログラム＝ひとつのプロセスということになっている。プログラムはHuman68kによって確保されたメモリにロードされ，諸々の管理情報を与えられた時点で，Human68kの管理下に置かれ，プロセスとなる。

Human68k上で一度に動けるプロセスはただひとつだけだが，動作中のプロセスは別のプロセスを生成・起動することができる。いわゆるチャイルドプロセスだ。あまり意識されないが，COMMAND.XやVS.X上から実行しているプログラムも，結局はみんなチャイルドプロセスである。

では，さっそく本題に入ろう。ちなみに今回の話の参考資料は『プログラマーズマニュアル』の第7章だ。

メモリ管理はOSの役目

メモリ上に複数のプロセスが共存するためには，ひとつの大前提がある。プロセス間でメモリの干渉が起きないよう，誰かがメモリの使用状況（どこが使用中で，どこが空いているのか）を把握・管理していなければならない。当然それはOSの役目だ。メモリの管理はOSに求められる最も基本的な機能といえる。

Human68kではメモリ管理ポインタと呼ばれる双方向リスト構造をしたデータを使ってメモリを管理している。簡単にいってしまうと，確保したメモリブロック（ひとかたまりのメモリ）の先頭にそのブロックの大きさや，前後のメモリブロックの位置を示す情報を付け加え，それによって管理するという方法だ。Human68kのメモリ管理ポインタは4ロングワード＝16バイトからなり，表1に示すような構造をしている（表中，Lはロングワードの意味）。メモリ管理ポインタは必ずxxxxx0$_H$のアドレスに置かれる。これは実質的にメモリ確保の最小単位が16バイトであることを意味する。

メモリ管理ポインタ内には直前と直後のメモリ管理ポインタのアドレスが格納されているから，これを利用して，メモリ管理ポインタを正逆両方向にたどることができる。a0レジスタにメモリ管理ポインタのアドレスが入っている状態から，

```
        movea.l (a0),a0
```

を実行すればa0は直前のメモリ管理ポインタを指し，

```
        movea.l 12(a0),a0
```

を実行すれば次のメモリ管理ポインタを指すようになる。

このメモリ管理ポインタのリンク（つながり）を

表1　メモリ管理ポインタ（計16バイト）

+00$_H$	1L	ひとつ前のメモリ管理ポインタのアドレス（0…前はない）
+04$_H$	1L	このメモリブロックを確保したプロセスのメモリ管理ポインタのアドレス（0…親はない）
+08$_H$	1L	このメモリブロックの終わり＋1のアドレス
+0C$_H$	1L	次のメモリ管理ポインタのアドレス（0…次はない）

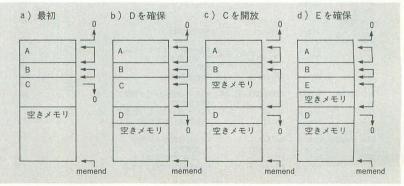



図1　メモリ管理のイメージ

直前，また直前というようにずっと前にたどっていくとOS内部にある最初のメモリ管理ポインタに突き当たる[1]。このメモリ管理ポインタはOSそのものが収められたメモリブロックを表していると考えられる。『プログラマーズマニュアル』の表記に従い，以下，この最初のメモリ管理ポインタの置かれたアドレスをfstmemと呼ぶことにする。なお，fstmemの位置はHuman68kのバージョンによって異なる[2]。

Human68kにおけるメモリ管理ポインタによるメモリ管理のイメージを図1に示す。最初のa)ではA～Cの3つのメモリブロックが確保されている。図1では特に明記していないが各メモリブロックの頭にはそれぞれメモリ管理ポインタがある。いま，それぞれのメモリ管理ポインタは，

0←A⟷B⟷C→0

というリンクを形成している（0はリンクの末端を示すものとする）。Cの後ろが空きメモリである。空きメモリはどのメモリ管理ポインタにも属さない部分であり，どこまでが空きメモリかは別の情報（図ではmemendとした）で示す必要がある[3]。

この状態から新たなメモリブロックDを確保するとb)のようになる。空きメモリの先頭を削ってDに割り当てた形だ。そののちDを開放すればa)に戻るわけだが，Dを確保したままでCだけを開放することもできる。c)の状態がそれで，BとC，CとDの間のリンクを断ち切って，つなぎ合わせた格好をしているのがわかる。

空いてしまったメモリは無駄にはならない。次に確保するメモリがここに収まる大きさであれば，d)のようにこの隙間を埋める形でメモリの確保が行われる。さらに残った隙間もまたあとで使われるチャンスがある。Human68kはメモリの確保を行うときに，必ず，メモリブロックの隙間に収まるかどうかをチェックする[4]。

ただ，Human68kの場合，隙間のチェックをメモリの低位から順に行うので，必ずしも最高の効率でメモリが使用できるわけではない。20Kバイト，10Kバイトの隙間がこの順序で存在するときに，10Kバイトのメモリを確保しようとすると，最初に見つけた20Kバイトの隙間を使ってしまい，10Kバイトの隙間2つが残るという結果になる。

メモリブロックの表示

メモリ管理ポインタは通常のプログラムでは特に気にする必要がないのだが，ついでなので，リスト1にメモリ管理ポインタのリンクをたどり，fstmem以降のメモリブロックのアドレスをすべて表示するプログラムMB.Xを示す。XCやHuman68k Ver.2.0などに付いているPROCESS.Xをずっと簡単にしたようなものだ（実行結果を見比べてみるとよい）。リスト2のITOH.Sとリンクして実行形式ファイルを作成するようになっている。ITOH.S中のサブルーチンitohは，以前作った数値→16進文字列変換サブルーチンを，スタックを介してパラメータを渡すように作り直したものだ。これからもちょくちょく

1) Human68kのメモリ管理ポインタは，必ずメモリの低位アドレスから順にリンクし，あっちいったりこっちいったりするようなことはない。

2) 『プログラマーズマニュアル』にはこのアドレスが，さも固定であるかのように具体的なアドレスで示してあるが無視すること（しかも示されているアドレスは間違っている）。

3) 余談ながら，Human68kのRAMディスクドライバはmemendに相当するHuman68k内部のワークエリアを操作することで，メインメモリの後ろからメモリを削り取って使っている。

4) 表1で確認してもらいたいのだが，"次のメモリ管理ポインタのアドレス"から"このメモリブロックの最終アドレス＋1"を引けば，隙間の大きさが求められる。

リスト1　MB.S

```
 1: *           メモリ上の全メモリブロックの位置を表示する
 2: *
 3: *           作成法：as mb
 4: *                   as itoh
 5: *                   lk mb itoh
 6: *
 7:             .include        doscall.mac
 8:             .include        const.h
 9: *
10:             .xref   itoh                    *外部参照
11: *
12: *           メモリ管理ポインタの構造
13: *
14: PREVMEM             equ     0
15: OWNERPROC           equ     4
16: MBEND               equ     8
17: NEXTMEM             equ     12
18: *
19:             .text
20:             .even
21: *
22: ent:
23:             lea.l   mysp,sp                 *spの初期化
24:
25:             clr.l   -(sp)                   *スーパーバイザモードに
26:             DOS     _SUPER                  *  切り換える
27:                                             *d0 = ssp
28:             move.l  d0,(sp)                 *sspを待避
29:
30: loop1:      move.l  PREVMEM(a0),d0          *d0 = 直前のメモリ管理ポインタ
31:             beq     loop2                   *0なら先頭
32:             movea.l d0,a0                   *a0 = 直前のメモリ管理ポインタ
33:             bra     loop1                   *先頭に達するまで繰り返す
34:
35: loop2:      pea.l   buff                    *メモリ管理ポインタの
36:             move.l  a0,-(sp)                *  先頭アドレスを
37:             bsr     itoh                    *  16進文字列変換して
38:             addq.l  #8,sp                   *  buff以降にセットする
39:
40:             move.b  #'-',buff+8             *つなぎの'-'を書き込む
41:
42:             pea.l   buff+9                  *メモリブロックの
43:             move.l  MBEND(a0),d0            *  最終アドレスを
44:             subq.l  #1,d0                   *  16進文字列に変換して
45:             move.l  d0,-(sp)                *  buff+9以降にセットする
46:             bsr     itoh                    *
47:             addq.l  #8,sp                   *
48:
49:             pea.l   buff                    *XXXXXXXX-XXXXXXXXまでを
50:             DOS     _PRINT                  *  まとめて表示する
51:             addq.l  #4,sp                   *
52:
53:             pea.l   crlfms                  *改行する
54:             DOS     _PRINT                  *
55:             addq.l  #4,sp                   *
56:
57:             move.l  NEXTMEM(a0),d0          *d0 = つぎのメモリ管理ポインタ
58:             movea.l d0,a0                   *←注意：フラグは変化しない
59:             bne     loop2                   *d0が0でなければ繰り返す
60:
61:             DOS     _SUPER                  *ユーザーモードに
62:             addq.l  #4,sp                   *  戻す
63:
64:             DOS     _EXIT                   *終了
65: *
66:             .data
67:             .even
68: *                    01234567890123456
69: buff:       .dc.b   '12345678-12345678',0   *表示用文字列格納領域
70: crlfms:     .dc.b   CR,LF,0                 *改行コード
71: *
72:             .stack
73:             .even
74: *
75: mystack:                                    *スタック領域
76:             .ds.l   1024                    *
77: mysp:
78:             .end    ent                     *実行開始アドレスはent
```

使うのでライブラリに加えておいてほしい。

MB.Xのアルゴリズムは，fstmemから順次，メモリブロックの先頭アドレスと最後のアドレスを表示してはリンクをたどる処理を繰り返すというだけの単純なものだ。最後のメモリブロックに到達したかどうかは，メモリ管理ポインタ内の“次のメモリ管理ポインタのアドレス”が0かどうかで判断する。ただ，fstmemが何番地か決められないため，メインの処理に先立ってfstmemの位置を探さなければならない。これはメモリ管理ポインタをメモリ低位方向へたどることで行う。

問題はもうひとつある。fstmemはOS内であり，ユーザー空間の外側のスーパーバイザ空間に位置している。通常のアプリケーションの走行環境であるユーザーモードではアクセスできない場所にあるわけだ[5]。そこで，一時的にスーパーバイザモードに移行する必要が出てくる。これにはDOSコールsuperを使う。リスト1の25行のようにスタックにロングワードで0を積んで[6]superを呼び出すとスーパーバイザモードに移行することができる。superから戻った時点でd0.lには直前までのssp（スーパーバイザスタックポインタ）が入っており[7]，これはあとでユーザーモードに戻るときに必要な情報なのでどこかに保存しておかなければならない。リスト1では素直にスタックにしまっている。

スーパーバイザモードに移行し，安心して悪さができるようになったところでメモリ管理ポインタをたどってfstmemを探す。プログラムが起動した時点でa0レジスタにはプログラムの置かれたメモリブロックのメモリ管理ポインタが入っていることになっているから，そこを基点にしてメモリ管理ポインタをひたすら前にたどっていく（30〜33行）。

fstmemが見つかったら，16進変換サブルーチンを適当に使ってメモリブロックの最初と最後のアドレスを表示する。ひとつのメモリ管理ポインタに対する処理がすんだら，リンクをたどって続くメモリブロックを処理する（35〜59行）。

もし続くメモリ管理ポインタがなければそれで処理はおしまいだ。終了直前に礼儀正しくスーパーバイザモードからユーザーモードに戻しておきたい。これはさっきスーパーバイザモードに移ったときにDOSコールsuperから戻ってきたsspの元の値をスタックに積んで再度superを実行することで行う[8]。

メモリの確保と解放

OSがメモリを管理している以上，ユーザープログラムはOSに与えられたメモリ以外に手を出してはいけない。もし，実行途中でより多くのメモリが必要になったらOSに要求し，確保してから使う。Human68kには，メモリ確保に関するDOSコールとしてmalloc, free, setblockの3つが用意されている。それぞれ，メモリブロックの確保，メモリブロックの解放，メモリブロックのサイズ変更を行う。

●malloc

mallocは次のようにして呼び出す。

```
move.l  確保したいバイト数,-(sp)
DOS     _MALLOC
addq.l  #4,sp
```

バイト数にメモリ管理ポインタの分を入れる必要はない。自動的にメモリ管理ポインタで使う16バイトだけ余計にメモリが確保される。また，バイト数は下位の24ビットのみが有効だ（上位8ビットは0と見なされる）。X68000のメインメモリは最大で12Mバイトだからこれで十分なのはいうまでもない。

メモリの確保に成功した場合はd0.lに確保して使えるようになったメモリのアドレスを入れて戻る。これはメモリ管理ポインタの直後にあたり，実際に使えるメモリの先頭アドレスだ。

メモリ不足で指定しただけのメモリが確保できなかった場合はd0.lに81000000_H＋確保できる最大のバイト数が返ってくる[9]。また，完全に確保不可能な

5）強引にアクセスしようとしても，ユーザーモードから見れば“そこにはメモリが実装されていないのと同じこと”なので，バスエラーが発生する。

6）実際にはclr.lを使っている。

7）すでにスーパーバイザモードなのに再びスーパーバイザに切り替えようとするとエラーになり，d0.lは負の値をとる。

8）リスト1ではいきなりsuperを呼び出しているように見えるが，28行ですでにsspをスタックに積んだ形になっているので，これでよいのだ。

9）mallocで一度に確保できる最大のバイト数は，空きメモリの合計ではなく，最大の大きさの空きメモリブロックの大きさであることに注意したい。

リスト2　ITOH.S

```
 1:         .xdef   itoh                *外部定義
 2: *
 3:         .text
 4:         .even
 5: *
 6: *itoh(value,buff)
 7: *機能： 32ビット整数を16進8桁の文字列に変換する
 8: *レジスタ破壊：ccr
 9: *
10: *       ex)
11: *               pea.l   buff
12: *               move.l  #$12345678,-(sp)
13: *               bsr     itoh
14: *               addq.l  #8,sp
15: *                   :
16: *       buff:   .ds.b   8+1
17: *
18: *
19: value   =       8
20: buff    =       12
21: *
22: itoh:
23:         link    a6,#0
24:         movem.l d0-d2/a0,-(sp)
25:
26:         move.l  value(a6),d0        *値
27:         movea.l buff(a6),a0         *文字列格納アドレス
28:
29:         moveq.l #8-1,d2             *以下を8回繰り返す
30:
31: itoh0:  rol.l   #4,d0               *d0.lを左に4ビット回転する
32:         move.b  d0,d1               *d0の下位バイトをd1に取り出し
33:         andi.b  #$0f,d1             *  下位4ビットを残してマスクする
34:         addi.b  #'0',d1             *ここで数値から16進を表す文字へ
35:         cmpi.b  #'9'+1,d1           *  変換する
36:         bcs     itoh1               *  0~9の場合は'0'を足すだけだが
37:         addq.b  #'A'-'0'-10,d1      *  A~Fの場合はさらに補正が必要
38:
39: itoh1:  move.b  d1,(a0)+            *変換した文字をしまう
40:
41:         dbra    d2,itoh0            *繰り返す
42:
43:         clr.b   (a0)                *文字列終端コードを書き込む
44:
45:         movem.l (sp)+,d0-d2/a0
46:         unlk    a6
47:         rts
```

場合は戻り値は$8200000x_H$になっている（xは不定だそうだ）。メモリ管理ポインタを格納するための16バイトすら取れなかったということだ。

どちらにしろ，$8xxxxxxx_H$は2の補数として見れば負の値であり，ほかのDOSコール同様，d0.lが負であればエラーと判断してよいことになる。

mallocの返すエラーを積極的に利用すると，リスト3[10)]のようなこともできる。ここに示すサブルーチンallmemは取れる限り大きなメモリブロックを確保し，その先頭アドレスをd0.lに入れて戻る。確保できなかったらd0.lは負の値になる。

allmemでは最初に絶対確保できない大きさのメモリを確保しようとする。当然エラーになるから，戻り値のd0.lは$81xxxxxx_H$か$8200000x_H$のどちらかだ。$81xxxxxx_H$だった場合は下位24ビットを残して上位ビットをマスクすると，確保できる最大バイト数が得られる。この値を使ってもう一度mallocを実行すれば，間違いなく最大の大きさのメモリブロックが確保できるわけだ。$8200000x_H$だった場合は，上位をマスクしてもう一度mallocしてもやはりエラーになるわけで，サブルーチンから戻った時点でメインルーチン側でエラーチェックすれば判別できる。

10）リスト3のALLMEM.SはITOH.S同様ライブラリとして使うためのものだから，単独で実行形式ファイルにしても意味がない。

●free

mallocで確保したメモリブロックを解放するにはfreeを使う。mallocの戻り値をスタックに積んで呼び出すだけだ。

```
move.l  確保したメモリのアドレス,－(sp)
DOS     _FREE
addq.l  #4,sp
```

変なアドレスを指定した場合など，エラーのときはd0.lに負の値を返すが，mallocの戻り値をそのまま使えば，エラーが起こることは考えられない。

なお，freeを使って明示的に解放しなくても，exitやexit2でプロセスが終了したときに，そのプロセスが確保したメモリはすべて解放されることになっている。だが，礼儀としては確保したメモリはきちんと解放してから終了すべきだろう。

●setblock

残るsetblockはmallocで確保したメモリブロックの大きさを変更するのに使う。小さくするのにも大きくするのにも使える。呼び出し方法は次のとおりだ。

```
move.l  変更したいバイト数,－(sp)
move.l  確保したメモリのアドレス,－(sp)
DOS     _SETBLOCK
addq.l  #8,sp
```

setblockの実行がうまくいけば，そのメモリブロックは指定した長さに変更されている。メモリブロックを小さくする分にはエラーは起きようもないが，大きくしようとしてできなかった場合はmalloc同様の値でd0.lにエラーを返してくる。

さて，Human68kのチャイルドプロセス起動時のメモリ割り当てには変な癖がある。最も大きな空きメモリブロックを全部確保して，プログラムを読み込む領域に割り当ててしまうのだ[11)]。メモリ上にHuman68kとCOMMAND.Xだけがある状態からユーザープログラムを起動すると，そのプログラムには最も大きな空きメモリブロック，つまり，フリーエリアの残り全部が割り当てられる。家を買ったら広い庭がオマケに付いてきたようなもので，プログラム本体はその頭の部分にちょこんと置かれた形になっている。この“庭”はmallocなんかしなくても，当然自由に使ってよい。すでに確保されているのだ。

11）リスト1のMB.Xを走らせた人なら，最後にやけに大きなメモリブロックがあるのに気づいたはずだ。

が，広いメモリを与えられたということは，そのメモリの管理がユーザープログラムに任されたことを意味する。どこを使っているのかを自分で把握している必要があるわけだ。OSの苦しみをいやというほど味あわされることになるだろう。そこで，いったん余分なメモリをOSに返却してしまい，それから必要に応じてmallocで確保して使うということがよく行われる。「こんなメモリはいらない」と戻しておいて，「でも，ちょっとちょうだいね」と少しずつ取り返すのだ。

余分なメモリの切り離しには，さっきのsetblockを使い，“プログラム自身が置かれているメモリブロック”を“プログラム本体を収めるぎりぎりの大きさ”に変更する。リスト4のサブルーチンmemoffがこの場合の常套的な手順だ。

setblockに渡すパラメータとして“プログラム自身が置かれているメモリブロックのメモリ管理ポイ

リスト3　ALLMEM.S

```
 1:         .include        doscall.mac
 2: *
 3:         .xdef   allmem          *外部定義
 4: *
 5:         .text
 6:         .even
 7: *
 8: *allmem()
 9: *機能： 確保できる最大のメモリブロックを確保する
10: *       d0.lに確保した先頭アドレスを持って戻る
11: *       d0.lが負の場合はエラー
12: *レジスタ破壊：d0.l,ccr
13: *
14: *       ex)
15: *               bsr     allmem
16: *               tst.l   d0
17: *               bmi     error
18: *
19: allmem:
20:         move.l  #$ffffff,-(sp)
21:         DOS     _MALLOC
22:         andi.l  #$ffffff,d0
23:         move.l  d0,(sp)
24:         DOS     _MALLOC
25:         addq.l  #4,sp
26:         rts
27:
28:         .end
```

ンタの直後のアドレス”と，“プログラム本体の大きさ”が必要だから，まず，これらを求める。前者はプログラム起動時のa0レジスタ（＝プログラムの置かれたメモリブロックのメモリ管理ポインタの先頭アドレス）に，メモリ管理ポインタで使っているメモリサイズ16を加えればよい。リスト４では15行の，

```
lea.l    16(a0),a0
```

でこのアドレスをa0に求めている。また，起動時のa1レジスタにはプログラム本体の終わり直後のアドレスが入っているので，a1からいま求めたa0の値を引けば，プログラムで使っているメモリの大きさが求まる。あとはこれらをsetblockに渡すだけだ。

プロセス管理とは

Human68kではプロセスは“プロセス管理ポインタ”[12)]と呼ばれる240バイトの構造体状のデータでプロセスを管理している。メモリ管理ポインタのように相互にリンクするようなことはなく，単に管理情報がひとまとめにしてあるだけのものだ。

プロセス管理ポインタは表２のような構造をしており（表中，L，W，Bはそれぞれロングワード，ワード，バイトを意味する），メモリ管理ポインタの直後に置かれる。慣例に従い，プロセス管理ポインタ内のデータはメモリ管理ポインタの先頭からの相対アドレス（アドレスの差）で示してある。プロセス管理ポインタの直後から実際のプログラムが格納される。プログラムの先頭にentというラベルを付けたとすると，プロセス管理ポインタの先頭はent－F0_Hで直接指定できる[13)]。

ここでは，プロセス管理ポインタの中身自体を理解する必要はない。「プログラムの頭にはこんな情報が付いているんだなー。プロセスを管理するのに必要そうな情報が並んでいるなー」という程度の認識で十分だ。実際，この内容を直接書き換えることはユーザープログラムには許されていないし，読んでみたところで役に立つような情報も少ない。

ただ，プロセス管理ポインタの80_H番地目以降に，起動されたプログラム自身のドライブ名，パス名，ファイル名が格納してあるのはちょっとおいしい。この情報はプログラムの実行に必要なデータファイルがある場合に重宝する。データファイルを実行ファイルと同じディレクトリに置いておくことにすれば，このプロセス管理ポインタ内の情報から，データファイルのあるディレクトリ（結局は実行ファイルのあるディレクトリ）がすぐ求められる。これで，“××というデータファイルのあるディレクトリに移動してから実行してください”などという間抜けなプログラムを作らなくてすむだろう。

●exec

プロセスに関するDOSコールはいくつかあるが，そのうち最も利用価値の高いのがプログラム中からほかのプログラムを起動するexecだ。このDOSコールひとつで子プロセスの起動が比較的簡単に行える。

なお，execで子プロセスを起動するときにはHuman68k内部でmallocが実行され，子プロセス用のメモリが確保される。このため，子プロセス起動の前に，さっきのmemoffのような処理で子プロセス用の空きメモリを作っておかなければならない。

execには０～４のモードが用意されており，モードによってパラメータの個数や意味が異なる。Human68k Ver.2.0では機能が若干拡張されているが，今回はVer.1.0と共通の部分だけを取り上げる。

モード0はプログラムのロード・実行を一気に行う。

```
move.l  環境変数領域のアドレス,－(sp)
move.l  コマンドラインパラメータ,－(sp)
move.l  起動ファイル名,－(sp)
clr.w   －(sp)              ＊モード０
DOS     _EXEC
lea.l   14(sp),sp
```

環境変数領域のアドレスに０を指定すると，子プロセスは親の環境をそのまま引き継ぐ。特別な環境で子プロセスを走らせようというのでない限り，いつも０にしておけばよいだろう。コマンドラインパラメータは任意の文字列（の先頭アドレス）で指定する。子プロセス側ではこのアドレスを起動時のa2レジスタで受け取ることになる。起動ファイル名はやはり適当な領域にファイル名をセットして，その先頭アドレスで指定する。最後にスタックに積んでいる０はモード番号の０だ。

リターン時のd0.lが負であればエラーが発生した（子プロセスが起動できなかった）ことを意味する。

12)『プログラマーズマニュアル』を見ると，プロセス管理ポインタの同義語として“PSP”，“PDB”という２つの略語が未定義のまま使われているようだ。前者はMS-DOSで使われている言葉で“Program Segment Prefix”の略だったと記憶している。“プログラムが格納された領域の頭書き”ぐらいの意味だ。後者はマニュアルではひと言も触れられていないが，きっと“Process (Program ?) Data Block”かなんかだろう（自信はないが）。

13) メモリ管理ポインタも同様にent－100_Hで指定できる。これはプログラム起動時のa0レジスタと同じ値になるはずである。

リスト４　MEMOFF.S

```
 1:         .include        doscall.mac
 2: *
 3:         .xdef   memoff              *外部定義
 4: *
 5:         .text
 6:         .even
 7: *
 8: *memoff()
 9: *機能： プログラム本体以降の余分なメモリを開放する
10: *           注意：プログラム起動直後、a0,a1が破壊される前に
11: *                 呼び出すこと
12: *レジスタ破壊：a0.l,a1.l,d0.l,ccr
13: *
14: memoff:
15:         lea.l   16(a0),a0           *a0 = メモリ管理ポインタの直後
16:                                     *a1 = プログラム本体の直後
17:         suba.l  a0,a1               *a1 - a0 = プログラムに必要な
18:                                     *          メモリサイズ
19:         move.l  a1,-(sp)            *メモリブロックの新サイズ
20:         move.l  a0,-(sp)            *メモリブロックの先頭アドレス
21:         DOS     _SETBLOCK           *メモリブロックサイズ変更
22:         addq.l  #8,sp               *スタック補正
23:
24:         rts
25:
26:         .end
```

d0.lが正であれば，それは子プロセスの終了コードである[14]。

モード1はロードのみを行い，実行アドレスをd0.lに返す。呼び出し時のパラメータはモード番号を除けばモード0と変わらない。

```
move.l 環境変数領域のアドレス,-(sp)
move.l コマンドラインパラメータ,-(sp)
move.l 起動ファイル名,-(sp)
move.w #1,-(sp)              *モード1
DOS    _EXEC
lea.l  14(sp),sp
```

モード1でロードしたのち実行するにはモード4を使う。モード1の戻り値をスタックに積んで渡すだけだ。

```
move.l 実行アドレス,-(sp)
move.w #4,-(sp)
DOS    _EXIT
addq.l #6,sp
```

モード2はモード0，1実行前の下準備に使う。具体的にはコマンドラインをファイル名とパラメータに分解し，同時に指定した環境変数領域内の環境変数pathを参照してファイルをディスクから検索する。ファイル名はこの検索結果に従って絶対パスで返される。

MORE CONFIG.SYS

のような文字列を，

A:¥BIN¥MORE.X
CONFIG.SYS

に分けてくれるわけだ。モード2の結果を使ってモード0か1を改めて実行すれば，pathが通っているディレクトリにあるプログラムならどれでも簡単にロード・実行できる。

```
move.l 環境変数領域のアドレス,-(sp)
move.l コマンドラインパラメータの格納
       領域,-(sp)
move.l 与えるコマンドライン兼ファイル
       名格納領域,-(sp)
move.w #2,-(sp)              *モード2
DOS    _EXEC
lea.l  14(sp),sp
```

環境変数領域はモード0，1同様，0を使った手抜き指定が可能だ。また，与えるコマンドラインがファイル名で上書きされる点には注意してもらいたい。

残るモード3はメモリの範囲を直接指定して，そこにプログラムをロードする。あまり使われないだろうからここでは相手にしない。X-BASICのFNCファイルの読み込みなんかに利用されているようだ。

子プロセスの起動

execはこのまま使ってもそれなりに便利なのだが，若干手間がかかるので，ここで，コマンドライン文字列を与えるとpathの検索からロード・実行まで行うようなサブルーチンを作っておく。サブルーチン名はchildとしよう。リスト5だ。大筋はexecのモード2と0を連続実行しているだけで，変なことはしていないつもりだから，しっかり読み切ってもらいたい。

まず，サブルーチンの頭で512バイトのローカルエリアを用意している[15]。execモード2で分解されるファイル名とコマンドラインパラメータを格納するのに256バイトずつ使う。リスト中ではそれぞれの先頭アドレスはスタックフレーム上の位置“nam buf(a6)”，“cmdlin(a6)”で表されている。以下，これらを単にnambuf，cmdlinと表記する。

ローカルエリア確保後，d0とspを除くレジスタをスタックに待避する[16]。それからサブルーチンに渡されたコマンドライン文字列の先頭アドレスをa0に取り出し，その文字列をnambufにコピーする。execモード2では与えた文字列を上書きする形でファイル名が返されるため，メインから渡された文字列を保存する意味でこういうことをしている。このコピーなら上書きされ失われても，メインルーチンに影響は出ない。

nambufへの文字列コピーは39～41行のループで行っている。nambufが256バイトしかないので，そ

14) 子プロセスの終了コードはDOSコールwaitを使っても取得できる。waitは単に，

DOS _WAIT

で呼び出し，最後に実行した子プロセスの終了コードをd0.lに返す。

15) childを呼び出すメインプログラム側では，この512バイトも計算に入れてスタック領域を確保する必要がある。

16) execで子プロセスを実行したときにはsp,sr以外のレジスタは破壊される。

表2 プロセス管理ポインタ

オフセット	サイズ	内容
●メモリ管理ポインタ		
+00H	4 L	
●プロセス管理ポインタ		
+10H	1 L	プロセスの環境変数領域アドレス（＝起動時のa3）
+14H	1 L	プロセスが終了した際の戻りアドレス
+18H	1 L	プロセスがCTRL+Cによって中断された際の戻りアドレス
+1CH	1 L	プロセスがエラーによって中断された際の戻りアドレス
+20H	1 L	プロセスに与えられたコマンドラインのアドレス（＝起動時のa2）
+24H	12B	プロセスのファイルハンドルの使用状況
+30H	1 L	プロセスのbssの先頭アドレス
+34H	1 L	プロセスのヒープの先頭アドレス（bssと同じ）
+38H	1 L	プロセスの初期スタックアドレス（＝プログラムの終わり＋1＝起動時のa1）
+3CH	1 L	親プロセスのuspの値
+40H	1 L	親プロセスのsspの値
+44H	1 W	親プロセスのsrの値
+46H	1 W	中止時のsrの値
+48H	1 L	中止時のsspの値
+4CH	5 L	trap10～14の例外ベクタ
+60H	1 L	プロセスのフラグ（0…親，－1…OSから起動された）
+64H	28B	未使用（Ver.2.0では一部使用）
+80H	1 L	このプロセスに対応するファイルのドライブ名（ドライブ名＋':'）
+82H	1 L	このプロセスに対応するファイルのパス名　（'¥～¥',0）
+C4H	1 L	このプロセスに対応するファイルのファイル名　（'～.X',0）
+DCH	36B	未使用
●プログラム本体		
+100H		～

れ以上は絶対に転送しないよう細工してある。

ここで使っているdbeqはdbraのバリエーションで，eqはbeqのeqと同じ“もしZフラグが立っていれば”という条件を表す[17]。dbeqは次のような動作をする。

1)　Zフラグが立っていれば，即座にループを抜け，dbeqの直後の命令の実行に移る
2)　そうでなければ，指定されたデータレジスタを1減らす
3)　結果が−1であればループを抜ける
4)　そうでなければ，指定アドレスに分岐し，ループを繰り返す

2)〜4)はdbraの動作と同じで，その前にフラグによる条件判断がついたものと思えばよい。感じとしては，

```
        beq     skip
        dbra    dn,~
skip:   ~
```

をひとまとめにしたようなものだ。“ある処理を一定回数繰り返すが，途中で特定の条件が成り立っていたらループを抜ける”のに便利な命令だ。ここでは“メモリ転送を255回繰り返すが，途中で文字列の終端コードを見つけたらそれ以上転送せずにループを抜ける”ために利用している。

nambufにコマンドライン文字列をセットしたら，以下，環境変数領域(手抜きの0)，cmdlin，nambufをスタックに積んで，execモード2，0を連続実行する。モード2の呼び出しでスタックに積んだパラメータがモード0でもそのまま使えるので，途中のスタック補正は省略し，スタックトップのモード番号を直接2から0に置き換えている（53行）。

execからの戻り値d0.lはそのままサブルーチンからの戻り値としてメインルーチンに返す。メイン側はd0.lの正負からエラーの有無を判定する。

childの使用例をリスト6に示す。このプログラムCHLDTEST.Xは単にチャイルドプロセスとしてATTRIB.Xを起動するだけのものだ。起動するコマンドはプログラム中に埋め込まれているので，適当に変更してみるとよい。メモリの切り離しにリスト4のmemoffを使っているから，実行ファイルは，

```
A>LK CHLDTEST CHILD MEMOFF
```

で作成すること。

さて，サブルーチンchildはCOMMAND.Xを経由することなく直接プログラムを子プロセスとして起動する。そのため，COMMAND.Xの内部コマンドや，リダイレクト，パイプなどの機能を利用することはできない。もし，これらの機能が使いたければ，

```
COMAMND DIR
COMMAND DIR >$$$
COMMAND DIR | MORE
```

のような文字列をchildに渡さなければならない。それが面倒であれば，

```
DIR | MORE
```

という文字列を渡すと，

```
COMMAND DIR | MORE
```

に変換してからchildを呼び出すようなサブルーチンを作っておけばよい。リスト7にその一例を示す。

256バイトのローカルエリアに56行で用意した“command ”という文字列＋メインから渡された文字列を作り上げ，childを呼び出すだけだ。プログラム上のテクニックとしては，2つのdbeqの微妙な使い方をチェックしておいてもらいたい。

17) 当然，dbneやdbccやdbcsなんかもある。

リスト5　CHILD.S

```
 1:         .include        doscall.mac
 2: *
 3:         .xdef   child                   *外部定義
 4: *
 5: *       EXECモード
 6: *
 7: LOADEXEC        equ     0
 8: PATHCHK         equ     2
 9: *
10:         .text
11:         .even
12: *
13: *child(cmd)
14: *機能： 与えられたコマンドラインに従って
15: *       プログラムをチャイルドプロセスとして起動する
16: *       d0.lにEXECの終了コードを持って戻る
17: *       d0.lが負の場合はエラー
18: *レジスタ破壊：d0.l,ccr
19: *
20: *       ex)
21: *               pea.l   cmd
22: *               bsr     child
23: *               addq.l  #4,sp
24: *               tst.l   d0
25: *               bmi     error
26: *                   :
27: *       cmd:    .dc.b   'command /cdir',0
28: *
29: nambuf  =       -512
30: cmdlin  =       -256
31: str     =       8
32: *
33: child:
34:         link    a6,#-512                *512バイトのローカルエリア
35:         movem.l d1-d7/a0-a6,-(sp)
36:
37:         movea.l str(a6),a1              *与えられた文字列を
38:         lea.l   nambuf(a6),a0           *  ローカルエリアに
39:         move.w  #255-1,d0               *  最大255バイト
40: chld0:  move.b  (a1)+,(a0)+             *  コピーしておく
41:         dbeq    d0,chld0                *∵上書きされるから
42:         clr.b   (a0)                    *念のための終端コード
43:
44:         clr.l   -(sp)                   *自分の環境
45:         pea.l   cmdlin(a6)              *パラメータ部格納領域
46:         pea.l   nambuf(a6)              *コマンドライン兼
47:                                         *  フルパス名格納領域
48:         move.w  #PATHCHK,-(sp)          *PATH検索
49:         DOS     _EXEC                   *
50:         tst.l   d0                      *d0.lが負なら
51:         bmi     chld1                   *  エラー
52:
53:         move.w  #LOADEXEC,(sp)          *ロード&実行
54:         DOS     _EXEC                   *
55: chld1:  lea     14(sp),sp               *スタック補正 4*3+2バイト
56:
57:         movem.l (sp)+,d1-d7/a0-a6
58:         unlk    a6
59:         rts
60:
61:         .end
```

18) その過程で困ったことにARG2.Sのバグを見つけてしまった。11月号P.111のリスト54行の，

```
	bsr getarg
```

は，

```
	bsr nextarg
```

の間違いだ。似たようなラベル名には気をつけよう（教訓になるなー）。

19) がーん。またもやバグを見つけてしまった。10月号P.62リスト6。268行の，

```
	bne	wopen0
```

は，

```
	bpl	wopen0
```

の間違いだ。と，このようにプログラムは使い回ししているうちにバグも取れ，だんだん信頼性が高くなるのだった（やー，教訓になるなあ）。

プロセス操作の応用

最後に，childを使った実用プログラムをひとつ示す。リスト8のRDERR.Xは“標準エラー出力をファイルにリダイレクトして，子プロセスを実行する”プログラムだ。エラーメッセージなど，標準エラー出力に出力されるメッセージをファイルに落としたいときに利用してもらいたい。

A>RDERR リダイレクト先ファイル コマンド

のようにして使用する。また，

A>RDERR /?

により，上記程度の簡単な使用法を標準エラー出力に出力して終了する。このあたりのコマンドラインパラメータ関係の処理は以前のARG2.Sから流用し，微調整してある[18]。

リダイレクトの処理にはDOSコールdup2を使っている。以前使ったdupはすでにオープンしたファイルハンドルをコピーして新しいファイルハンドルを返すものだったが，dup2はコピー先のファイルハンドル番号を指定できる。つまり，ファイルハンドルを強制的に（上書きする形で）コピーするわけだ。ファイルハンドルAとBがあるときに，dup2でAをBに強制コピーすると，AとBは同じファイルを指すようになる。A＝あるファイルをオープンしてできたファイルハンドル，B＝標準エラー出力のファイルハンドルと考えれば，これは正しく標準エラー出力のリダイレクションである。心配なのはこのファイルハンドルが子プロセスに引き継がれるかどうかだが，標準入出力やエラー出力に使われる0～4のファイルハンドルはちゃんと引き継がれることになっている。

プログラム中では50行以下がリダイレクトの処理だ。まず，すでにコマンドライン解析ルーチンによってfilnam以下に切り出されているリダイレクト先ファイルをcreateで新規作成する。以前のUPPER.Xを参考に，createでエラーが出ても，もう一度openで書き込みモードでオープンしている。これはconなどのキャラクタデバイスが指定された場合に備えてのことだった[19]。

リスト6 CHLDTEST.S

```
 1: *		チャイルドプロセスを単に起動してみる
 2: *
 3: *		作成法：as chldtest
 4: *			lk chldtest child memoff
 5: *
 6:		.include	doscall.mac
 7:		.include	const.h
 8: *
 9:		.xref	child		*外部参照
10:		.xref	memoff		*
11: *
12:		.text
13:		.even
14: *
15: ent:
16:		lea.l	mysp,sp		*spの初期化
17:
18:		bsr	memoff		*余分なメモリを開放する
19:
20:		pea.l	cmd		*チャイルドプロセス起動
21:		bsr	child		*
22:		addq.l	#4,sp		*
23:		tst.l	d0		*エラー？
24:		bmi	error		* そうならエラー終了
25:
26:		DOS	_EXIT		*終了
27: *
28: error:					*エラー終了
29:		pea	errmes
30:		DOS	_PRINT
31:		addq.l	#4,sp
32:
33:		move.w	#1,-(sp)
34:		DOS	_EXIT2
35: *
36:		.data
37:		.even
38: *
39: cmd:		.dc.b	'attrib *.*',0		*起動コマンド
40: errmes:	.dc.b	'コマンドが起動できません',0
41: *
42:		.stack
43:		.even
44: *
45: mystack:				*スタック領域
46:		.ds.l	1024
47: mysp:
48:
49:		.end	ent		*実行開始アドレスはent
```

リスト7 COMMAND.S

```
 1:		.include	doscall.mac
 2: *
 3:		.xdef	command		*外部定義
 4:		.xref	child		*外部参照
 5: *
 6:		.text
 7:		.even
 8: *
 9: *command(cmd)
10: *機能： command.xをチャイルドプロセスとして起動する
11: *		d0.lにEXECの終了コードを持って戻る
12: *		d0.lが負の場合はエラー
13: *レジスタ破壊：d0.l,ccr
14: *
15: *		ex)
16: *			pea.l	cmd
17: *			bsr	command
18: *			addq.l	#4,sp
19: *			tst.l	d0
20: *			bmi	error
21: *				:
22: *		cmd:	.dc.b	'dir | more',0
23: *
24: temp		=	-256
25: str		=	8
26: *
27: command:
28:		link	a6,#-256
29:		movem.l	a0-a1,-(sp)
30:
31:		lea.l	temp(a6),a0	*一時領域に
32:		lea.l	comstr,a1	* 'command 'の文字列を
33:		move.w	#255-1,d0	* コピーする
34: com0:	move.b	(a1)+,(a0)+	*
35:		dbeq	d0,com0		*
36:
37:		subq.l	#1,a0		*a0は行き過ぎている
38:
39:		movea.l	str(a6),a1	*与えられた文字列を
40: com1:	move.b	(a1)+,(a0)+	* それに連結する
41:		dbeq	d0,com1		*（合計で255バイトまで）
42:
43:		clr.b	(a0)		*念のための終端コード
44:
45:		pea.l	temp(a6)	*"command "+strを
46:		bsr	child		* 実行する
47:		addq.l	#4,sp		*
48:
49:		movem.l	(sp)+,a0-a1
50:		unlk	a6
51:		rts
52: *
53:		.data
54:		.even
55: *
56: comstr:	.dc.b	'command ',0
57:
58:		.end
```

create，openともにエラーであれば，エラーメッセージを出して終了する。オープンできたらそのファイルハンドルを標準エラー出力を表すファイルハンドルにdup2でコピーし，いまオープンしたファイルハンドルは不要になったのでクローズする。で，メインルーチンに戻る。

メインルーチンではその後，childを呼び出して子プロセスを起動し，子プロセスから戻ったら，リダイレクトされていた標準エラー出力をクローズして元のconに戻す。childの戻り値（＝execの戻り値）を調べてエラーの有無を判別する前に標準エラー出力を閉じているのは，エラーメッセージを標準エラー出力に出している関係だ。順序を間違えると，エラーメッセージがリダイレクト先のファイルに行ってしまい，エラーメッセージを標準エラー出力に出している意味がなくなる。

＊

と，今月の話はこれまで。

プロセス関連はつつくとなかなか面白く（人にもよるだろうが），プロセス管理ポインタ周りではトリッキーな技もかなりある。妙な（決して役には立たない）サンプルプログラムも何本か用意してあったのだが，肥大化した本文に押し出されて闇に葬られてしまった，残念。ま，読者諸氏も，暴走を顧みずいろいろ試してみるとOSやマシン語に対する知識の幅を広げることができるかもしれない。暗黒の迷宮であるOS内を懐中電灯片手にうろつき回るのもまた一興である。

ところで，当連載でこれまでに紹介した68000の命令，Human68kのDOSコールの種類は結構な数になる。主要なところは押さえたようだ。次回はそのほかの命令やDOSコールを（全部というわけではないが）ひととおりさらってみたい。

では，また来月。

リスト8　RDERR.S

```
  1: *          標準エラー出力をリダイレクトする
  2: *
  3: *          作成法：as rderr
  4: *                  lk rderr child memoff
  5: *
  6:            .include        doscall.mac
  7:            .include        const.h
  8: *
  9:            .xref   child           *外部参照
 10:            .xref   memoff          *
 11: *
 12:            .text
 13:            .even
 14: *
 15: ent:
 16:            lea.l   mysp,sp         *spの初期化
 17:
 18:            bsr     memoff          *余分なメモリを開放する
 19:
 20:            bsr     chkarg          *コマンドラインの解析
 21:
 22:            bsr     do              *メイン処理
 23:
 24:            DOS     _EXIT           *終了
 25:
 26: *
 27: *          メイン処理
 28: *
 29: do:
 30:            bsr     err_redirect    *標準エラー出力を
 31:                                    *  リダイレクト
 32:            move.l  a2,-(sp)        *チャイルドプロセス起動
 33:            bsr     child           *
 34:            addq.l  #4,sp           *
 35:
 36:            move.l  d0,-(sp)        *EXECのエラーコードを退避
 37:
 38:            move.w  #STDERR,-(sp)   *標準エラー出力をクローズ
 39:            DOS     _CLOSE          *（割り当てはconに戻る）
 40:            addq.l  #2,sp           *
 41:
 42:            move.l  (sp)+,d0        *d0.l=EXECの終了コード
 43:            bmi     error2          *負ならエラー
 44:
 45:            rts
 46:
 47: *
 48: *          標準エラー出力をfilnamにリダイレクトする
 49: *
 50: err_redirect:
 51: wopen:
 52:            move.w  #ARCHIVE,-(sp)  *指定されたファイルを
 53:            pea.l   filnam          *  新規作成する
 54:            DOS     _CREATE         *
 55:            addq.l  #6,sp           *
 56:            tst.l   d0              *エラー？
 57:            bpl     wopen0          *  エラーがなければオープン完了
 58:
 59:            move.w  #WOPEN,-(sp)    *createでエラーが発生したときは
 60:            pea.l   filnam          *  openを使って
 61:            DOS     _OPEN           *  もう一度ライトオープンしてみる
 62:            addq.l  #6,sp           *
 63:            tst.l   d0              *エラー？
 64:            bmi     error1          *  そうなら今度こそエラー終了
 65:
 66: wopen0:    move.w  d0,d1           *d1.w=出力先ファイルハンドル
 67:
 68:            move.w  #STDERR,-(sp)   *オープンしたファイルハンドルを
 69:            move.w  d1,-(sp)        *  標準エラー出力に
 70:            DOS     _DUP2           *  強制コピー
 71:            addq.l  #4,sp           *
 72:            tst.l   d0              *エラー？
 73:            bmi     error1          *  そうならエラー終了
 74:
 75:            move.w  d1,-(sp)        *いまオープンしたファイルハンドルは
 76:            DOS     _CLOSE          *  もういらないから
 77:            addq.l  #2,sp           *  クローズしてしまう
 78:
 79:            rts
 80:
 81: *
 82: *          コマンドラインの解析
 83: *
 84: chkarg:
 85:            addq.l  #1,a2           *a2=コマンド行文字列先頭
 86:
 87:            bsr     nextarg         *スペースをスキップする
 88:            tst.b   (a2)            *パラメータがあるか？
 89:            beq     usage           *  ないならパラメータが足らない
 90:            lea.l   filnam,a0       *a0=ファイル名切り出し領域
 91:            bsr     getarg          *パラメータ１つをa0以降に取り出す
 92:
 93:            lea.l   -100(sp),sp     *100バイトのローカルエリアに
 94:            move.l  sp,-(sp)        *  DOSコールを使って
 95:            move.l  a0,-(sp)        *  ファイル名を
 96:            DOS     _NAMECK         *  展開してみる
 97:            lea.l   100+8(sp),sp    *
 98:            tst.l   d0              *d0が０でなければ
 99:            bne     usage           *  ファイル名の指定に誤りがある
100:
101:            bsr     nextarg         *さらにスペースを飛ばす
102:            tst.b   (a2)            *まだあるか？
103:            beq     usage           *  実行すべきコマンドがない
104:
105:            rts
106:
107: *
108: *          スペースを飛ばしつぎのパラメータ先頭までポインタを進める
109: *
110: nextarg:
111:            bsr     skipsp          *スペースをスキップ
112:
113:            cmpi.b  #'/',(a2)       *パラメータの先頭が
114:            beq     usage           *  /,-であれば
115:            cmpi.b  #'-',(a2)       *  使用法を表示
116:            beq     usage           *
117:
118:            rts
119:
120: *
121: *          a2の指す位置からパラメータ１つ分を
122: *          a0の指す領域へコピーする
```

```
123: *
124: getarg:
125:           move.l  a0,-(sp)        *{レジスタ待避
126: gtarg0: tst.b   (a2)            *1)文字列の終端コードか
127:           beq     gtarg1          *
128:           cmpi.b  #SPACE,(a2)     *2)スペースか
129:           beq     gtarg1          *
130:           cmpi.b  #TAB,(a2)       *3)タブか
131:           beq     gtarg1          *
132:           cmpi.b  #'-',(a2)       *4)ハイフンか
133:           beq     gtarg1          *
134:           cmpi.b  #'/',(a2)       *5)スラッシュ
135:           beq     gtarg1          *
136:           move.b  (a2)+,(a0)+     * が現れるまで転送を
137:           bra     gtarg0          * 繰り返す
138: gtarg1: clr.b   (a0)            *文字列終端コードを書き込む
139:           movea.l (sp)+,a0        *} レジスタ復帰
140:           rts
141:
142: *
143: *         コマンド行先頭のスペースをスキップする
144: *
145: skpsp0: addq.l  #1,a2           *ポインタを進め
146:                                 *繰り返す
147: skipsp:                         *サブルーチンはここから始まる
148:           cmpi.b  #SPACE,(a2)     *スペースか?
149:           beq     skpsp0          * そうなら飛ばす
150:           cmpi.b  #TAB,(a2)       *TABか?
151:           beq     skpsp0          * そうなら飛ばす
152:           rts
153:
154:           DOS     _EXIT
155:
156: *
157: *         使用法の表示&終了・エラー終了
158: *
159: usage:
160:           lea     usgmes,a0
161:           bra     err0
162: *
163: error1:
164:           lea     errms1,a0       *ファイルオープン時エラー
165:           bra     err0
166:
167: error2:
168:           lea     errms2,a0       *チャイルドプロセス生成時エラー
169:           bra     err0
170:           *
171: err0:   move.w  #STDERR,-(sp)   *標準エラー出力へ
172:           move.l  a0,-(sp)        * メッセージを
173:           DOS     _FPUTS          * 出力する
174:           addq.l  #6,sp           *
175:
176:           move.w  #1,-(sp)        *終了コード1を持って
177:           DOS     _EXIT2          * エラー終了
178:
179: *
180: *         データ
181: *
182:           .data
183:           .even
184: *
185: usgmes: .dc.b   '機 能:標準エラー出力をファイルに切り換えてから',CR,LF
186:           .dc.b   '        指定のコマンドを実行します',CR,LF
187:           .dc.b   '使用法:RDERR 切り換え先ファイル 実行コマンド'
188: crlfms: .dc.b   CR,LF,0
189: errms1: .dc.b   'ファイルが作成できませんでした',0
190: errms2: .dc.b   'コマンドが起動できませんでした',0
191:
192: *
193: *         ワークエリア
194: *
195:           .bss
196:           .even
197: *
198: wfno:   .ds.w   1               *リダイレクト先ファイルハンドル
199: filnam: .ds.b   256             *ファイル名切り出し用バッファ
200: *
201:           .stack
202:           .even
203: *
204: mystack:                        *スタック領域
205:           .ds.l   1024            *
206: mysp:
207:
208:           .end    ent             *実行開始アドレスはent
```

思考よ〜ん(その1)

Iwai Ippei
満開製作所 祝 一平

先月は休講したC調言語講座。今月は思考ルーチンであるミニマックス法についての講義です。ミニマックス法とはチェスゲームなどでお馴染みの指手先読みのワザ。「ぽよよ〜ん」で始まる2カ月ぶりの一平氏は気合いが入ってます。今月は珍しく文章が多いけど，しっかりと理解しましょう。

ぽよよ〜ん。最近ディープソートというチェスマシンが新聞や雑誌で評判になっているみたいだけど，着実にコンピュータも思考を深めとるわけやね。詳しいことはよく知らないけど，なんでもカスタムチップを使っているそうで，このチップの性能のおかげで，初めて人間のグランドマスター（強さに対する称号）に勝てたのだそうだ。

思考ルーチンの話

さて，この思考ルーチンだけど，オセロあたりではかなりのところまできているそうな（それでもやっぱりまだ人間のチャンピオンには勝てないそうだけど）。あと，森田の将棋とかもわりといいとこまできているようですな。しかし，私の勝手な考えでは，いちばん厳しいのは囲碁だと思う。

いまの思考ルーチンの最大の問題点は学習機能の弱さであろうと思う。つまり，一度犯した間違いを，また繰り返す（というよりも，自分の間違いを発見/抽出できない）という欠点があるわけだな。たとえば極端な話，「同じ間違いは絶対二度と繰り返さない」という思考プログラムができれば（相当メモリを喰いそうだな），そいつは数多く対戦を繰り返すことによって，少しずつではあるにせよ，確実に強くなっていくわけでしょ。要するに，いまの技術はそこまできていないわけだ。

そして，最近情けなく思うのがシミュレーションゲームの思考ルーチンである。別にものすごく強くなくてもいいから，少なくとも「油断できない」というレベルにまで達してくれないだろうか(もちろんズルなしで)。ここだけの話だが，古典的ベストセラーの戦国シミュレーションゲームで，コンピュータ側のレベルを上げてプレイしていた人が，たまたま操作ミスで，コンピュータ側が扱っていたあるひとつの領地を，1ターン内で2度のぞいたことがあったそうな。で，その結果わかったことは，その思考ルーチンは，1ターンのうちにズルして何度も動くことによって，強くなっていたということであった。ちゃんちゃん。

しかしそれでさえも，いまの思考ルーチンは，まだまだマヌいのである。言ってみれば，「いよっ。若旦那，お強いですねぇ。さすがっ！ もう，あたしゃコテンパンでゲス」なんていう，タイコモチ的思考ルーチンであるような気がする。そして，そんな思考ルーチンに勝ったぐらいで，小鼻をぴくぴくさせて喜んでいる人も多いようである。

勝負ごとは，なんでも勝ち負けがあるわけだ。でもって，勝てば気持ちいいけど，負けるとプリプリなわけである。だから，負けたほうは強くなろうと努力したりするのである。しかし，そのいやな負け役を，学習能力と向上心のほとんどないパソコンが受け持ってくれたりすると，プレイヤーは馬鹿旦那に成り果てたりするわけである。接待プログラムの功罪や如何。

大分前に読んだイギリス人の文章であるが，「スポーツの最大の効用は，負けを教えてくれることだ」というのがあった。一種の精神修養というとらえ方であるな。確かに負け惜しみとか，言い逃れは紳士的じゃなく，見苦しいもんな。というわけで，いまのパソコンゲームのような，タコい思考ルーチンに勝ち続けていると，変に自信を持っちゃって，負け惜しみの強い，いやーな人間に育ってしまうかもしれないよ〜ん。

というわけで，パソコン少年の精神修養のためにも，ときには人間に勝つような（もちろんズルなしで）思考ルーチンがあったらいいなぁと思うわけである。さもないと，結局は思い込みの強い人しか楽しめないだけのものになってしまうかもしれないぞぇ。

思考のイロハ

コンピュータにゲームをさせる場合，もしも「読み切り」ができればそれですべてが解決するのであるが，たいていのゲームではそれは無理というものである。なにせ，起こりうるゲームの局面の数が「全宇宙の粒子の数よりも多い」なんてのはザラだからである。たとえば，大雑把に言って，囲碁であれば19×19の階乗，すなわち361！の局面があるのだ（実際はそれよりもちょっと少ないが，まあ，似たようなもの）。

そこで，この状況を克服するために評価関数というものを持ち出すのである。これは，盤面の形勢を数値化したもので，たとえばいい局面なら30点，負けそうな局面なら－50点とかの点数をつけるのである。で，あとはこの点数ができるだけ高くなるように駒を動かしていけばいいわけだな。

この関数の完璧なものがわかるということは，すなわち，最善手がわかるということなわけで，となれば完璧なプレイが可能に

なるのである。しかし当然ながら，世の中はそう甘くないのである。実際は，「まあ，だいたいは当たっているだろう」という点数しか計算できないのである。

そこで，少しでもその近似の精度を上げるために，ミニマックスという手法を使うのであった。ここらへんはゲーム理論の基礎である。

ゲーム理論

ゲーム理論というのは，みんなもよく知っているフォン・ノイマンおじさんが，40年くらい前に，もうひとりのお友達と一緒に考え出したものである。よーするに基本的にこの何十年の間進歩していなかったりする。

まず評価関数を使って駒を動かす方法で，いちばん簡単なのは一手読みである。これはすべての動きを試してみて，そのなかで評価関数の点数が最大になるものを選ぶのである。これはかなりもっともな方法であろう。しかし，問題は先ほども述べたように評価関数が完璧じゃないということなわけだ。つまり，一見するといい手に見えるが，どっこい，チャンスの裏にピンチありの言葉が示すように，実際はその手を指すと，何手か先に引っくり返される運命にあるかもしれないわけである。これは特にオセロゲームなんかではよく出くわすことで，「やった！ 隅を取ったぞ！ うぴうぴ」と思ったのもつかの間，何手かあとにはそのかいもなくボロボロに負けてしまって悔しい思いをした人も多いはずだ。

そこで，考え出されたのがミニマックス理論である。

これは簡単に言ってしまえば，いまから何手か先の評価関数の示す点数を予測しよう/引っくり返されるかもしれない事態を前もって予測しよう，ということである。

ミニマックス理論とは

まずはわかりやすいように，いちばん浅いミニマックスを考えてみよう。ゲームは，まあ，なんでもよいのだが，とりあえずは白対黒で，スペースの都合もあるから，それぞれの「手」は常に3通りあって，そのうちのひとつを選ぶことによってゲームが進んでいくということにする。で，評価関数は常に白にとっての点数ということにしよう。すなわち，白はこの数字を大きくしたいし，黒はこの数字を小さくしたいのである。まずは最初に，評価関数の状態が図1のようだとしよう。

で，ここに表示されている範囲内で考えるなら（つまり白も黒もこれより深くは考えないと仮定する），もしも白がA案を採用したなら，きっと黒はそれに対してaを選択するだろうということがわかる。なぜなら，黒はできるだけ小さい数字にしたいのだから，a＝1，b＝3，c＝2のうちからならば，aの1を選ぶはずである。で，Aの手における評価関数は3だけど，黒がそれに対抗する手を考慮に入れると，実はAの本当の点数は，3ではなく1だというほうが正確らしいということがわかるのである（あくまでこの枝振りの様子から考えれば，ということを忘れずに)。同じように，もしも白がB案を採用したなら，黒の手はきっとbで，結果は3。もしも白がC案なら，黒はきっとbで，結果は2であろう。こう考えると，評価関数の数字では，

A → 3
B → 5
C → 9

だったのであるが，黒のリアクションまでも考えに入れると，

A → 1
B → 3
C → 2

という点数のほうがもっともらしく見えてくるのである。そこで，白はこの数字をできるだけ大きくしたいわけだから，B＝3を選ぶべきだということになるわけである。

この過程をもう一度たどってみる。

黒は数字を小さくしようとするので，A－x，B－x，C－xのうちから，もっとも小さなものを選ぼうとする。結局黒が選ぶであろうと思われるのは，

$$\min_{x}(A-x)=A-a=1$$

$$\min_{x}(B-x)=B-b=3$$

$$\min_{x}(C-x)=C-b=2$$

の3つである。

で，白は数字を大きくしたいわけであるから，これらのうちからもっとも大きいB－b=3を選ぶわけである。で，これは，

$$B-b=\max_{X}(\min_{x}(X-x))$$

図1

図2

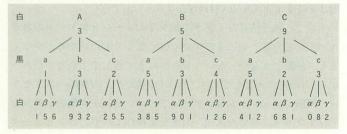
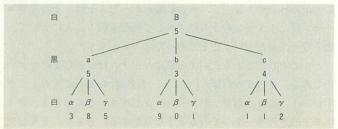
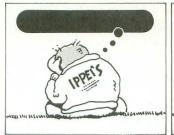



図3

なわけなのだ。評価値の最小（ミニマム）を選んだなかから，最大（マキシマム）を選び出す。これがミニマックスの意味なのである。

ここで理解しておかなければならないのは，これはあくまでゲームの展開をモデル化したものであって，現実の勝負ごとは，こんなに単純ではないということである（本質の部分では，通い合うものがあることはあるけど)。

上の例では白がBを選べば，黒はできるだけ頑張ってbを選ぶということになっているが，実際は黒には黒の考えがあるのだから，白の予測どおりに動くとは全然限らない。

さらには，たとえば，もう一手深く読むと，実は図2のようになっているのかもしれないのだ。

で，たとえば白がBを選んだとする。黒はそれに対して，図3にクローズアップしたような枝振りにくるわけである。

ここでミニマックスを適用するのであれば，黒は逆の立場だからマックスをとってからミニを取るので，

$a-\beta=8$

$b-\alpha=9$

$c-\gamma=2$

となる。これを最小にしたいわけだから，$c-\gamma=2$が狙い目となる。すなわち，黒は白のBに対して，白の予想であるbではなく，cを選ぶのである。そうなると白は「ラッキー！　3点に行き着くと思ってたけど，黒は4点の枝を選んだぞ」と考えるわけであるが，黒としては「$a-\beta=8$や$b-\alpha=9$を回避して$c-\gamma=2$で収めたぞ，うしししし」と思っているのである。

なんかお互いの思惑がチグハグだけど，これがミニマックスの実体である。

ミニマックスの欠点

「自分の評価関数は十分に正しい」

という，かなり強引な前提のもとに，

「おそらく敵も，自分のと同じような評価関数を使っているに違いない」

とみなすわけである。かなり無謀な手法であるが，さりとてほかに「これだ！」というのも見つからないので，まあ，「やっちゃえ，やっちゃえ！」なのである。

しかし，やっぱりなかには王より飛車をかわいがる輩もいるわけで，そのようなときは自分の評価関数と，敵の行動予測用の評価関数を，別々に使う必要が出てくるのであるが，ま，そこらへんはマイナーなアレンジにすぎないから，おいておく。

ミニマックスでは，このように交錯した思惑のなかでゲームが進んでいくわけである。基本原理はそれほど面倒なものではないからゆっくり考えれば問題ないであろう。

さて、このミニマックスの最大の欠点というのが，たとえ簡単なゲームでも，数段の探索でたちまち枝分かれがトンでもなく多くなるということである。すなわち計算時間がやたらとかかるのである。そうなると，本当は5段くらい先までの枝振りを見たいところを，我慢して3段ぐらいであきらめるということになる。その結果，先のほうがよくわかんないので，あまり強い思考ルーチンにできないということになってしまうのである。

で，「ここから先の枝振りは見る必要はない」というのを判定し，少しでも計算時間の短縮をもくろむのが$\alpha-\beta$切断である。そしてこれは来月のテーマである。

プログラムの解説

ちょっとでも複雑なゲームになると，たちまち枝がブハブハに繁ってしまうので，あれこれ考えたのだがなかなかよいサンプルが見つからなかった。

そこで，「1列だけの蛙跳びゲーム」というのをやってみること

図4　プログラム実行結果

```
A>g1 0 0
WHITE = 0,  BLACK = 0
W         Www000bbb --> 0wwW00bbb    0
B         0www00bBb --> 0www0Bb0b    7
W         0Www0bb0b --> 00wwWbb0b    1
B         00wwwbB0b --> 0Bwwwb00b    6
W         0bWwwb00b --> 0b0wwbW0b    2
B         0b0wwBw0b --> 0bBww0w0b    5
W         0bbww0W0b --> 0bbww00Wb    6
B         0bbww00wB --> 0bbww0Bw0    8
W         0bbww0bW0 --> 0bbww0b0W    7
B         0bbww0B0w --> 0bbwwB00w    6
W         0bbwWb00w --> 0bbw0bW0w    4
B         0Bbw0bw0w --> B0bw0bw0w    1
W         b0bw0bW0w --> b0bw0b0Ww    6
B         b0Bw0b0ww --> bB0w0b0ww    2
W         bb0W0b0ww --> bb00Wb0ww    3
B         bb00wB0ww --> bb0Bw00ww    5
W         bb0bW00ww --> bb0b0W0ww    4
B         bb0B0w0ww --> bbB00w0ww    3

BLACK WIN

A>g1 1 0
WHITE = 1,  BLACK = 0
W         Www000bbb --> 0wwW00bbb    0
B         0www00bBb --> 0www0Bb0b    7
W         0Www0bb0b --> 00wwWbb0b    1
B         00wwwbB0b --> 0Bwwwb00b    6
W         0bWwwb00b --> 0b0wwbW0b    2
B         0b0wwBw0b --> 0bBww0w0b    5
W         0bbww0W0b --> 0bbww00Wb    6
B         0bbww00wB --> 0bbww0Bw0    8
W         0bbww0bW0 --> 0bbww0b0W    7
B         0bbww0B0w --> 0bbwwB00w    6
W         0bbWwb00w --> 0bb0wbW0w    3
B         0Bb0wbw0w --> B0b0wbw0w    1
W         b0b0Wbw0w --> b0b00bwWw    4

WHITE WIN
```

にした。で，この最大の欠点は，なにげなく作った評価関数が，どうやら「完璧」だったらしくて，わざわざミニマックスをしなくても1段だけ見て，評価関数を最大にする駒の動きを選べばそれで最善手になってしまったということである。

「う～む。これではサンプルにならない」と思ったが，そこはそれ，転んでもタダでは起きない私である。さっそく評価関数を，ワザと少しおかしくしてみて，「ほうら，このように評価関数がおかしくても，ミニマックスを使えばちゃんといい手を指すようになるんだよ」と，開き直ってしまうのである。

元の「完璧らしい評価関数」は，/＊～＊/で注釈文にして無効にしてあるvtableという配列（表）をもとに計算してある（リスト1)。そのあとに作ったワザとおかしくした評価関数用の表と見比べていただきたい。

で，ゲームというのは9マスの左端3マスには白，右端3マスには黒がいる。白は右へ，黒は左へ移動するのだが，前に石がいなければ1マス進むだけ。もしも前に石があり，そしてその先に(何個先でもよい)に空いたマスがあるのなら，そこへぴょんと跳んでいける。ただし1回に一度しか飛べない。もしも空きがなかったら，その石は動けない（3個とも動けなかったらパス)。そして，動ける石がある以上はパスは許されない(リスト2)。

以上がルールである。で，プログラムの実行には，

g1　白のレベル　黒のレベル

の形式で白と黒の思考レベルを指定する必要がある。その結果であるが，

白＝0，黒＝0　→　18手目で黒の勝ち

白＝1，黒＝0　→　13手目で白の勝ち

白＝1，黒＝1　→　16手目で黒の勝ち

などとなっている。図4の実行結果を参照のこと。

行く年来る年

前述のように，来月は枝切りの基本の$\alpha-\beta$切断をやるつもりである。できればある程度遊べるゲームも用意したいと思うが，なにせTRONがすぐそこまできているらしいので，1990年の1月にはニュージーランドに亡命しているかもしれない(ウソ)。では，よいお年を。

リスト1　インクルードファイル(g1.h)

```
================  G1.H  ==================
   1: #define SIZE 9              /* 盤の大きさ */
   2: #define KOMA 3              /* 駒の数 */
   3:
   4: #define EMPTY 0             /* 空き */
   5: #define WHITE 1             /* 白駒 */
   6: #define BLACK 2             /* 黒駒 */
   7:
   8: #define   BIGNUM  9999      /* 無効（ありえない値） */
   9: #define   MIN(x,y)          (((x)<(y)) ? (x):(y))
  10:
  11: /*        元の表
  12: int vtable[3][SIZE] = {
  13:     0, 0, 0, 0, 0, 0, 0, 0, 0,
  14:     0, 1, 2, 3, 4, 5, 6, 6, 6,
  15:    -6,-6,-6,-5,-4,-3,-2,-1, 0};
  16: */
  17: int vtable[3][SIZE] = {
  18:     0, 0, 0, 0, 0, 0, 0, 0, 0,
  19:     0, 0, 1, 3, 2, 1, 6, 6, 6,
  20:    -6,-6,-6,-1,-2,-3,-1, 0, 0};
  21:
  22: UBYTE board[SIZE];             /* 盤 */
  23: int WHITE_P,BLACK_P;           /* 終了判定用変数（フラグ） */
  24:
  25:
```

リスト2　1列だけの蛙跳びゲーム(g1.c)

```
================  G1.C  ==================
   1: #include  <class.h>
   2: #include  <stdio.h>
   3: #include  "g1.h"
   4:
   5: main(argc,argv)
   6: int argc;
   7: char *argv[];
   8: {
   9:   int i,level0,level1;
  10:   int number;               /* 動かすべき駒番号 */
  11:   auto int vt[SIZE];        /* 評価値 */
  12:
  13:   level0 = level1 = 0;
  14:   if (argc > 1) {           /*level0 = 第1引数 = 白のレベル*/
  15:           level0 = atoi(argv[1]);
  16:   }
  17:   if (argc > 2) {           /*level1 = 第2引数 = 黒のレベル*/
  18:           level1 = atoi(argv[2]);
  19:   }                         /* 思考レベルの設定 */
  20:
  21:   printf("WHITE = %d,  BLACK = %d¥n",level0,level1);/* レベル表示 */
  22:   init_board(board);        /* 盤の初期化 */
  23:
  24:   while(1) {
  25:           for(i=0;i<SIZE;i++) {   /* 評価表を作る */
  26:                   vt[i] = think(board,i,WHITE,level0);
  27:           }
  28:           number = max(vt);        /* 最大値を与える位置を求める */
  29:           display_move(board,number);    /* 駒の動きを表示する */
  30:           if (gameend(board)) {   /* ゲームエンド? */
  31:                   break;
  32:           }
  33:
  34:           for(i=0;i<SIZE;i++) {   /* 評価表を作る */
  35:                   vt[i] = think(board,i,BLACK,level1);
  36:           }
  37:           number = min(vt);        /* 最小値を与える位置を求める */
  38:           display_move(board,number);    /* 駒の動きを表示する */
  39:           if (gameend(board)) {   /* ゲームエンド? */
  40:                   break;
  41:           }
  42:   }
  43: }
  44:
  45: /* levelの深さまで探索する */
  46: int think(p,n,color,level)
  47: UBYTE *p;
  48: int n;
  49: UBYTE color;
  50: int level;
  51: {
  52:   int i,j,number;
  53:   int v,table[SIZE];
  54:   UBYTE vb[SIZE];
  55:   UBYTE rcolor;
  56:
  57:   if (p[n] != color) {
  58:           return(BIGNUM); /* 対象外なら無効 */
  59:   }
  60:
  61:   if (level <= 0) {
  62:           return(test_eval(p,n));
  63:   }               /* 動かしてみた場合の評価値だけを返す */
  64:
  65:   rcolor = rev(color);     /* 敵の色 */
  66:   for(i=0;i<SIZE;i++) {
  67:           for(j=0;j<SIZE;j++) {
  68:                   vb[j] = p[j];
  69:           }               /* 一時的に盤の状態をコピー */
  70:           if (move(vb,n) < 0) {
  71:                   v = BIGNUM;      /* 動けないなら無効 */
  72:           } else {         /* 1レベル下げて敵の手を探る */
  73:                   v =think(vb,i,rcolor,level-1);
  74:                   if (v == BIGNUM) {
  75:                           v = eval(vb);
```

```
 76:           }/* 無効なら動かしてみた場合の評価値だけを返す */
 77:           }
 78:           table[i] = v;    /* 評価値をテーブルに */
 79:   }
 80:
 81:   if (color == WHITE) {
 82:           i = max(table); /* 白なら最大にしたがる */
 83:   } else {
 84:           i = min(table); /* 黒なら最小にしたがる */
 85:   }
 86:   if (i >= 0 ) {
 87:           return(table[i]);          /* これが最終的な評価値 */
 88:   } else {
 89:           return(BIGNUM); /* 最大／最小がない */
 90:   }
 91: }
 92:
 93: /* 黒←→白 */
 94: rev(color)
 95: UBYTE color;
 96: {
 97:   return((color == WHITE) ? BLACK : WHITE);
 98: }
 99:
100: /* 最小の評価値の駒番号を返す */
101: int min(table)
102: int *table;
103: {
104:   int i,n,v,v0;
105:
106:   n = -1;
107:   v = BIGNUM;     /* DUMMY */
108:   for(i=0;i<SIZE;i++) {
109:           if ((v > (v0 = table[i])) && (v0 != BIGNUM)) {
110:                   v = v0;
111:                   n = i;
112:           }
113:   }
114:   return(n);
115: }
116:
117: /* 最大の評価値の駒番号を返す */
118: int max(table)
119: int *table;
120: {
121:   int i,n,v,v0;
122:
123:   n = -1;
124:   v = -BIGNUM;    /* DUMMY */
125:   for(i=0;i<SIZE;i++) {
126:           if ((v < (v0 = table[i])) && (v0 != BIGNUM)) {
127:                   v = v0;
128:                   n = i;
129:           }
130:   }
131:   return(n);
132: }
133:
134: /* 駒の動きを表示する */
135: display_move(p,n)
136: UBYTE *p;
137: int n;
138: {
139:   int m;
140:
141:   printf("%c¥t","0WB"[p[n]]);/* 白黒の表示(最近知ったテク)*/
142:   display(p,n);                          /* 元を表示 */
143:   if (n >= 0)  {
144:           m = move(p,n);                 /* 動かす */
145:           printf(" --> ");
146:           display(p,m);                  /* 移動後を表示 */
147:           printf("    %d¥n",n);
148:   } else {
149:           printf("¥tPASS¥n");
150:   }
151: }
152:
153: /* 盤bのn盤目の駒を動かした場合の評価を返す */
154: /* 実際には動かさない */
155: int test_eval(b,n)
156: UBYTE *b;
157: int n;
158: {
159:   UBYTE t[SIZE];
160:   int i,v;
161:
162:   for(i=0;i<SIZE;i++) {
163:           t[i] = *b++;     /* copy */
164:   }
165:
166:   if (move(t,n) >= 0) {    /* 動かす */
167:           v = eval(t);     /* 評価 */
168:   } else {
169:           v = BIGNUM;      /* 無効値 */
170:   }
171:   return(v);
172: }
173:
174: /* 盤pのn盤目の駒を動かす */
175: /* 動けないなら-1を返す */
176: int move(p,n)
177: UBYTE *p;
178: int n;
179: {
180:   int i,v;
181:
182:   v = -1;
183:
184:   switch(p[n]) {
185:   case EMPTY:
186:           break;
187:
188:   case WHITE:
189:           for(i=n+1;i<SIZE;i++) {
190:                   if (p[i] == EMPTY) {      /* 跳び超える？ */
191:                           p[n] = EMPTY;
192:                           p[i] = WHITE;     /* 着地 */
193:                           v = i;
194:                           break;
195:                   }
196:           }
197:           break;  /* 動けなかった */
198:   case BLACK:
199:           for(i=n-1;i>=0;i--) {
200:                   if (p[i] == EMPTY) {      /* 跳び超える？ */
201:                           p[n] = EMPTY;
202:                           p[i] = BLACK;     /* 着地 */
203:                           v = i;
204:                           break;
205:                   }
206:           }
207:           break;  /* 動けなかった */
208:   }
209:   return(v);
210: }
211:
212: /* 盤を表示 */
213: display(p,n)
214: UBYTE *p;
215: int n;
216: {
217:   int i;
218:
219:   for(i=0;i<SIZE;i++) {
220:   if (i == n) {
221:                   printf("%c","0WB"[*p++]);
222:           } else {
223:                   printf("%c","0wb"[*p++]);
224:           }
225:   }
226: }
227:
228: /* 評価関数の本体 */
229: int eval(b)
230: UBYTE *b;
231: {
232:   int i,val;
233:
234:   WHITE_P = BLACK_P = val = 0;
235:   for(i=0;i<SIZE;i++) {
236:           val += vtable[b[i]][i];
237:           if (b[i] == WHITE) {
238:                   WHITE_P += vtable[WHITE][i];     /* 終了判定用 */
239:           } else {
240:                   BLACK_P -= vtable[b[i]][i];      /* 終了判定用 */
241:           }
242:   }
243:   return(val);
244: }
245:
246: /* 駒の初期化 */
247: init_board(p)
248: UBYTE *p;
249: {
250:   int i;
251:
252:   for(i=0;i<SIZE;i++) {
253:           *p++ = EMPTY;
254:   }
255:   /* 駒を初期状態に置く */
256:   for(i=0;i<KOMA;i++) {
257:           board[i] = WHITE;
258:           board[SIZE-i-1] = BLACK;
259:   }
260: }
261:
262: /* ゲームの終了をチェック */
263: int gameend(p)
264: UBYTE *p;
265: {
266:   eval(p);
267:   if (WHITE_P == 18) {
268:           printf("¥nWHITE WIN¥n¥n");
269:           return(1);
270:   }
271:   if (BLACK_P == 18) {
272:           printf("¥nBLACK WIN¥n¥n");
273:           return(1);
274:   }
275:   return(0);
276: }
```

脳ミソどろどろフレームソース

プロジェクトチーム DōGA かまた ゆたか
三保　陽介

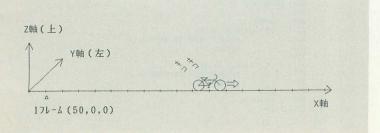
動きのデザインの2回目ということで，フレームソースファイルの書式と作り方について勉強しましょう。マニュアルにも結構詳しく書いてありますが，実例が乏しいので，より具体的に解説します。今回はちょっと手ごわいぞ。

少し前の話になりますが，本誌11月号でCGAシステムの配布終了をほのめかしたら，あわてた読者からの応募が来るわ来るわ。1日当たりの申し込み数の記録を，軽く2倍ほど更新する日が連日続きました。とても処理できたものではない。ちゃんとギャグを入れている方はほとんどいないし（Mさん，あなたのギャグはヒンシュクをかいました。当チームには女性のスタッフもいるのですよ。ここに内容を掲載できないような下ネタは困ります），しばらく受付が放棄されてしまいました。気を取り直して，受付を再開して気がついたのですが，“マニュアルが足りない！”

よっぽどお断り申し上げようかとも考えたのですが，断るだけでも大変な量なので，しかたないので，この時期になって増刷に踏み切りることになりました。ということで，増刷分のマニュアルがなくなるまではCGAシステムの申し込みを受け付けております。皆さんふるってご応募ください。ハッハッハッ。

フレームソースとは

フレームソースファイルは，動きに関するデータファイルです。“動き”というのは，“時間（フレーム）が変化するときに位置が変化すること”ですから，“1フレーム目に（50,0,0）にある物体が，20フレーム目には（1000,0,0）に移動している”というふうに表現することができます。そのとき，“1フレーム目は（50,0,0）で，2フレーム目は（100,0,0），3フレーム目は（150,0,0）で，4フレーム目は……”といちいち指定してやるのはめんどうなので，1フレーム目と20フレーム目の位置を指定して，その間は均等補間してやるのが普通ですし，均等補間でなくてもいろいろな式や関数を用いて表現することもできます（図1）。

図1
座標軸と向き（右手系）

“動き”は位置の変化とは限りません。物体の大きさや向きの変化もあります。たとえば，Z軸方向に小さくしてやれば，ペシャンコにつぶれていく“動き”になります。それに，動くものは物体とも限りません。カメラ（視点）が動くのはCGでよく用いる手法ですし，光の当たる方向だけが“動く”アニメーションだってあります。極端な話，光の色が変わっていくのも，ある種の“動き”といえます。

さらにフレームソースでは，動かないもの（“動き”が0）も一緒に取り扱います。ですから，フレームソースは，“動きのデータ”というよりも，物体を配置し，光と視点を用意する，つまり“空間のデザイン”といったほうがよいかもしれません。

先月の宿題の答え合わせ

さて，皆さん宿題はちゃんとやってきましたか？（忘れた人は廊下に立ってなさい）。

まず，例1をご覧ください。これは，動きに関するデータを実際にFFEでデザインしたフレームソースそのものです。これをこれから解説するのですが，この形式

例1　FFEの出力ファイル

```
#frame( fno, 1, 20 )
@4.2@
env { back ( rgb ( 0.00 0.50 0.80 ) ) }

fram
{
        light pal( rgb ( 1.00 1.00 1.00 ) -3.00 -2.00 -4.00 )
        { mov (      500
                     200
                     400
              )
              eye deg( 60 )
        }
        { mov ( ¥div(         50,    1000,
                               1,      20,fno )¥
                     0
                     0
              )        target
        }
        { mov ( ¥div(         50,    1000,
                               1,      20,fno )¥
                     0
                     0
              )
              obj    BICYCLE
        }
}
#endframe
```

で書いていたのでは，やたらに改行が多く，ページを無駄に使ってしまうので，本文においては例２のように記述します。

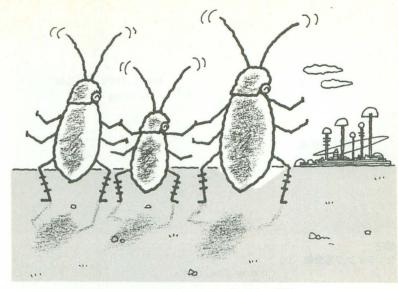
フレームソースにおいては，スペース，改行，タブは同じ区切り記号の意味しか持ちませんので，例１と例２はまったく同じということになります。また，例２以降の各リストの左端に付いている数字は解説のための行番号で，実際のフレームソースには付きません。

フレームソースを記述するときの注意のひとつとして，コマンドなどはすべて小文字でなければいけません。物体名だけは大文字でもかまいませんが，これはCAD（モデリングツール）でセーブした名前と合わせないと「オブジェクト名 ＊＊＊＊ は登録されていません。」と文句をいわれてしまいます。本文では，区別しやすいように物体名はすべて大文字を使用するとしましょう。

それでは，「例２」を解説しましょう。まず１行目では，フレームソースの始まりを宣言しています。10行目は対になっている終わりの宣言です。この手の宣言はおまじないですので，なにも考えずにフレームソースの最初と最後に付ければよいわけです。ただ，１行目の括弧の中は，“fno という変数が１から20まで変化する20フレーム分のフレームソースである”という意味があります。BASICの，

FOR fno=1 TO 20

と同じようなものです。ですから，(fno, 50, 100) という記述もできます。

２行目は“動き”をいろいろな式や関数を用いて表現する際の精度を設定しています。この例では，４桁の精度で小数点以下が２桁ということです。整数を出力してやりたければ，@4.0@とします。また，ひとつのフレームソースの中で何度でも設定し直すことができますから，ここは整数で，ここは６桁で，などと細かい指定をすることができます。しかしこの行はなくてもかまわないので，あまり気にする必要はありません。

３行目は環境文，４～９行目がフレーム文です。環境文とは，作画する前の画面の状態，いわゆる背景設定です。この例では，べた塗り(back)で色は空色(rgb (0.00 0.50 0.80))となっています。rgbの値に式を使うこともでき，

env { back (rgb (¥fno/20¥ 0.0 0.0)) }

とすると，(フレームごとに fno の値が増えていくので)背景がだんだん赤く染まっていくといった表現もできます。また，環境文を省略すると，背景がない，つまり透

```
 1: #frame ( fno, 1, 20 )
 2: @4.2@
 3: env  {  back (  rgb ( 0.00 0.50 0.80 ) ) }
 4: fram {
 5:      light pal( rgb ( 1.00 1.00 1.00 ) -3.00 -2.00 -4.00 )
 6:      { mov ( 500 200 400 )          eye deg( 60 )   }
 7:      { mov ( ¥div( 50, 1000, 1, 20, fno )¥ 0 0 ) target }
 8:      { mov ( ¥div( 50, 1000, 1, 20, fno )¥ 0 0 ) obj  BICYCLE }
 9: }
10: #endframe
```

例２
ＦＦＥの出力ファイル
（改訂版）

アマチュアCGAコンテスト事務局より

も～いくつ寝るとCGAコンテストの締め切りです。ラストスパートかかっていますか？ 作品が思ったほど集まらないのではないかと，夜も眠れぬ日々が続いています。ぜひ皆さんの積極的なご参加お願いいたします。

第２回「アマチュアCGAコンテスト」応募要項

●募集作品

パーソナルコンピュータを使用したアマチュアのオリジナル作品

- 実写が含まれていてもかまわない
- DōGA・CGAシステムを使用している必要はまったくない
- 静止画は基本的に不可
- プロの方でも，プライベートに制作したものなら可

●締切

1989年12月31日

●応募方法

- DōGA・CGAシステムのマニュアルに付いている応募用紙に必要事項を記入し，作品（コピーでよい）と一緒にコンテスト事務局までお送りください。
- 応募用紙をご希望の方は，事務局まで至急ご連絡ください。

〒533 大阪市東淀川区淡路5-17-24 篤コーポ２号館102号室
CGAコンテスト事務局

●応募形態

VHS，S-VHSが望ましい。8mmフィルムでも可

●読者特別救済処置

“イメージユニットがないので，VTRに落とせない”，“完成が間に合わない”などのお便りをいただきました。そこで（応募作品も少なくて困っていることですし），以下のような救済処置を行います。

・フロッピーディスクでの応募可

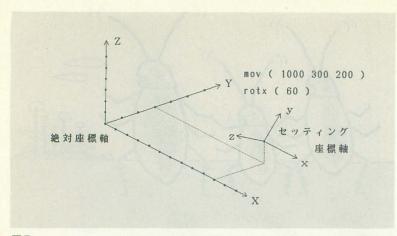
お送りいただいたFDからVTRに録画させる作業を代行します。もちろんDōGA・CGAシステムによる作品に限ります（静止画をたくさん送ってきて，8mmでコマ撮りしろったってお断りします）。当チームのスタッフが正月休みを返上して作業することになるのですから，録画しやすいようによく整理して，詳しく解説も同封することをお忘れなく（ディスクを入れたらオートスタートぐらいが望ましい）。ややこしい作業を要求したり，わけがわからなかったりしたら，その時点で選外（論外ともいう）とさせていただきます。

また，脈絡のないアニメーションが数カットあるというだけでは作品といえません。少なくともタイトルぐらいは付けてください。

・未完成，予告編あり

除夜の鐘の音を聴きながら，「うぉーとうとう間に合わなかったっ！」という方へ。まだ完成していない作品，予告編でも，一応エントリーしてください。多少不利にはなりますが，ちゃんと審査いたします。また審査の途中で，完成作品と交換することも可能です。

図2
セッティング座標軸

```
 1: #frame ( fno, 1, 20 )
 2: fram {
 3:      light pal( rgb ( 1.00 1.00 1.00 ) -3.00 -2.00 -4.00 )
 4:      { mov ( ¥fno*20¥ 200 50 )      eye deg( 60 )  }
 5:      {
 6:           mov ( 100 0 0 )      obj TREE
 7:           mov ( 100 0 0 )      obj TREE       target
 8:           mov ( 100 0 0 )      obj TREE
 9:      }
10: }
11: #endframe
```

例3 movの使い方

明の状態になります。

5行目では光の設定を行っています。この行を日本語にすると，“光（light）は平行光線（pal）で，色は白，ベクトル（−3,−2,−4）の方向に照らしている”といった具合になります。

6行目は視点の設定です。視点は，原点から（500,200,400）だけ移動した位置で画角が60度となっています。視線はこの視点の位置と7行目のターゲット（注目点）の位置を結んだ線になります。しかし，7行目のターゲット指定の文では，movの中に式が書かれている，つまりターゲットの位置が移動しますので，視線もそれに合わせて動くことになります。

8行目に物体（obj）の指定として「BICYCLE」があるわけですが，これも移動します。よく見ると，ターゲットの動きと同じですので，視点は常に「BICYCLE」を見ていることになります。

さてフレームソースでは，¥と¥に囲まれている部分は，式または関数と見なされ，計算を行います。環境文の解説例でもあったように，式としては通常の四則演算のほか，括弧や sin，log，累乗など，皆さんがご存じの関数はたいてい使えます。また，C言語のような記述の方法でユーザーが関数を定義することもできます。この例では div というフレームソースが独自に持っている関数が用いられています。div は補間という意味で，この例では，「fno が1のときに50,20のときに1000となる」ように，つまり ¥fno＊50¥ と同じ意味になります。

movで平行移動

まずmovですが，mov(X Y Z)という形式で，X，Y，Z方向の平行移動量を表します。ここで注意しなければいけないことは，mov(X Y Z)は(X，Y，Z)という位置を表しているのではなく，あくまでも移動量なのです。では，何が移動しているのでしょう？

ここで，「セッティング座標軸」というものを仮想します（名前はいまこの場で命名したいいかげんなものだ）。このセッティング座標軸は，最初は絶対座標軸と重なっているのですが，mov（平行移動），rot（回転移動）などのコマンドがあると移動します（図2）。そして，obj，eye, target など位置を設定するコマンドがあると，現在セッティング座標軸がある位置に，向きに，設定するわけです。

例3をご覧ください。3本の「TREE」を植えていますが，決して同じ位置に重なっているわけではありません。1本目は（100,0,0）の位置にあります。2本目は，7行

寺田の教育的指導

さて，今月登場する作品は，以前この連載でも紹介した「グラフィック研究会」の「あがた」氏の作品「ごきぶりさん」です。ごきぶりに「さん」を付けてしまうことでもわかるように，この作品は「あがた」氏の「ごきぶりさん」に対する愛情にあふれています。まず背景をご覧ください。ごきぶりにとって理想の環境（？）ともいえるパステルカラーに統一された空間に，パステルカラーのオブジェを配置するという心憎さ。「ごきぶりさん」に対する深い配慮の表れといえるでしょう。この素晴らしい環境に放たれたごきぶりの，喜びにあふれた縦横無尽の走りを見てください。ごきぶりの喜びが画面から伝わってきます。特に足の動きがごきぶりらしい雰囲気を出しています。よく見ると，1本1本の足は単純な形でできており，その動きも簡単な4パターンの繰り返しだけなのですが，「ごそごそ」と音でも聞こえてきそうな実感がありますね。このデフォルメのテクニックは大いに参考にしましょう。

さすがGRさんだけあって，アトリビュートの使い方，アンチエリアシングなど，この連載で取り上げたようなポイントはちゃんと押さえてます。そのうえ最終画面と最初の画面を同じにしておいて，1カットがループになっているなど，デモとして心憎い気配りです。せっかくここまで作ったのですから，あとは1本の作品としてまとめていけばよいと思います。背景をコンデンサやトランジスタにして，回路上を走り回らせたあと，ひとつのLSIの中に入っていくと，LSIのたくさんの足から，分裂した「ごきぶりさん」が四方八方に走り去っていく……なんてストーリーはいかがでしょう。CGAコンテストもありますし，頑張ってください。

今月はもう1本ご紹介しましょう。今度は趣向が変わって，NHKの放送コンクール応募用に作られた，北海道の追分高校放送部の「青い空とぼくたちの地球」です。公害問題をテーマとして取り上げた実写も含めた作品で，これなどCGAシステムのよい利用法といえるでしょう。では特別ゲストとして放送部OBでもある三保氏にも意見を聞きたいと思います。

寺田（以下て）：実は私も元放送部員なんですが，どうですか感想は？
三保（以下み）：ナレーションがしっかりしていて放送部ならではの出来ですね。
て：ナレーションがいいと作品が引き締まりますからね。
み：CGAを要所要所に効果的に使っているのがいいですね。
て：かっこいいじゃないですか。オゾンや酸性雨を取り上げたテーマもタイムリーですし。
み：CGA自体の技術レベルはあまり高くはないのですが，全体的に見れば，積極的にCGAを取り入れた意欲的な作品だと思うのですが。
て：確かにCGA作品としては，いま一歩といった感もありますが，1本の作品としてまとまっているところは，さすが映像制作になれた放送部といった感じです。放送コンクールのニューウェーブですね。

いかがでしたか。皆さんも，作品ができたら送ってくださいね。お待ちしています。それではまた，次回に。失礼します。

目の mov によって，セッティング座標が，1本目の「TREE」の位置からX軸方向に100だけ移動していますので，(200,0,0) ということになります。3本目は，同様に (300,0,0) です。さて，7行目の target ですが，2本目の「TREE」を植えたあと target まで移動を示すコマンドがありません。つまりセッティング座標軸はそのままの状態で位置を設定するのですから，2本目の「TREE」のあるところを注目していることになります。

rotで座標軸の回転

次に rot ですが，これはセッティング座標軸をセッティング座標軸まわりに回転させます。rotx，roty，rotz の3種類があり，当然X，Y，Z軸まわりの回転を表しています。単位は「度」で，回転の＋－は，その軸の＋の方向へネジが進むように回すのを＋とします。また，3次元空間における軸回転は，X，Y，Zの順番が異なると結果が異なるので注意が必要です。つまり，

rotx (30) roty (−90) rotz (20) obj BICYCLE

と，

rotz (20) rotx (30) roty (−90) obj BICYCLE

とは同じ向きにならないということです（頭がウニになりそう）。

例題1：原点から出発して，以下のように移動した場合「TREE」はどこに，どのような向きに立っているでしょうか。

```
mov  ( 500 0 0 )
rotz ( −90 )
mov  ( −500 0 0 )
roty ( 90 )
obj  TREE
```

実際にこのような記述は，フレームソースの文法上なんら問題ありません。正解は図3にあります（ほ～ら，頭が納豆になってきた）。

ポイントは2回目の平行移動 mov (−500 0 0) です。セッティング座標軸が回転しているので，セッティング座標のｘ軸と絶対座標のX軸は異なる方向を向いているのです。図4のような間違いを起こさないように。

図4 例題1の誤答例

例4 例題2の解答

```
 1: #frame ( fno, 1, 36 )
 2: @4.2@
 3: env  {  back ( rgb ( 0.0 0.2 0.6 ) ) }
 4: fram {
 5:         light pal ( rgb ( 0.5 0.7 1.0 ) 3 6 -8 )
 6:         { mov ( -900 0 100 )    eye deg ( 70 ) }
 7:         { mov (  500 0 -400 )   target }
 8:         {
 9:             mov (  500 0 -200 )
10:             rotz ( ¥fno*10¥ )
11:             mov (  0 -800 0 )
12:             obj   SAKANA
13:         }
14: }
15: #endframe
```

図3 例題1の解答

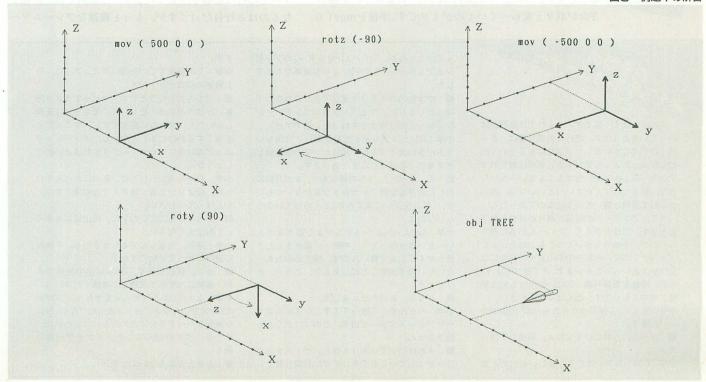
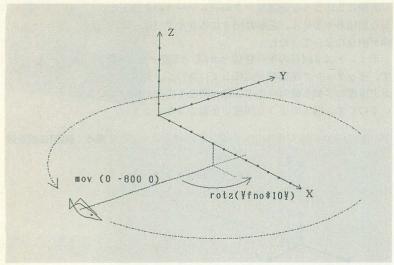



図5
例題2の解説図

同様に2回目の回転 roty （ 90 ） のY軸とはセッティング座標のｙ軸です。

それでは次に実戦的な応用問題です。かなり難しいですよ。

例題2：sin や cos を用いずに,obj SAKANA を正確な円運動させよ。ただし，中心は（500,0,−200），半径は 800 ，回転はZ軸まわりに＋の向きとする。

正解は，例4のようになります。よく見て理解できれば十分です。sin，cos を使うと，誤差がありますし，この方法では回転の角度を与えるのが簡単なうえ，自動的に進行方向に頭を向けるなどのメリットがあります。

まず9行目で，回転の中心を与えています。そして10行目で回転の角度を与えます(10度から360度まで10度ずつ)。11行目は半径になります。図5をご覧ください。mov （ 0 −800 0 ）というのは，ｙ軸方向に −800 動かすことですが， rotz （ ¥fno＊10¥ ） によって，ｙ軸の方向が刻々と変わっていくのがミソです。半径をmov(0 800 0 ）としてしまうと，「SAKANA」が後ろ向きに泳いでしまいますので，注意してください。

scalで大きさを変える

scal （ X Y Z ）という形式で各軸方向の大きさを変化させます。X，Y，Zの値が1以下の場合は小さくなりますし，負の値ならその軸方向に反転します。しかし，0であってはいけません。

この scal で注意することは，物体の大きさを変化させるというより，セッティング座標軸自体を変化させます。つまり，scal によってｘ方向の大きさを2倍にしたあとで mov をすると，ｘ方向に2倍移動します（脳みそを溶かさないように)。

例5と図6をご覧ください。「その1」が scal を使用していないときで，「CAT」が3匹並んでいます。「その2」では，scal の後ろには obj CAT があるだけですので，3匹目の「CAT」だけが2倍になります。しかし，この例のように，単に物体だけを巨大化させると，物体同士が交差することがあります。「その3」のように，mov の前に scal があると，移動量も大きくなるため，相対的な位置関係を崩さないのです。

もちろんこの scal は，rot に対しては影響がありません。それから余談になりますが，この例5では target がありませんが，間違いではありません。target がないときは，視線はｘ軸の正の方向（前）となります。

｛と｝で有効範囲を限定

例5「scalの使い方 その3」をもう一度ご覧ください。7行目にある scal が，7行目だけにとどまらず以下すべての行に影響を与えています。この例では，問題となるのは8行目だけですが，もっと複雑なフレームソー

あき姫の迷える小羊のコーナー

こんにちは！ 姫です。ちまたでは風邪がはやっているようです。流行に敏感な姫は，さっそくひいてしまいました（以前にもこういう出だしがあったような気がする)。皆様はお元気ですか？ 夜ふかしがいちばんよくないみたいなので，気をつけてください（といいつつ，夜ふかしして原稿を書いている姫なのであった)。

さて，アンケートの回答が続々と寄せられております。力作が多くて，たいへんおもしろかったので，そのうちのいくつかをご紹介しましょう（ところで，今回の小羊の方々のなかには女性が3名いらっしゃいます。さて誰でしょう)。

小羊：珍格言集番外編 DōGA の沙汰も腕次第。
姫：そのとおりです。がんばってください。
小羊：宝くじ，1億円当たったら100万円追加カンパします。
姫：う〜ん，うれしいですねぇ。期待して待ってます。
小羊：CGAシステムはとても難しいものだと思っていましたが，「パソコンサンデー」のプレゼントでもらった「R-TYPE」よりも簡単なものでした。
姫：中学生の方でもひと通りできたとわかってほっとしました。でも「パソコンサンデー」がなくなったのは残念ですね（古い話ですが)。
小羊：CGAシステムがまだ着かないと苦情をいう人が多いようですが，私はかつてある通販ビデオを約2年待ったことがあります。
姫：え，えらい！ うちの妹なんか，1カ月前に出した応募者全員プレゼントが届かないぐらいで，「私は全プレにはずれた」とぐちっているぐらいなのに。
小羊：DōGA・CGA システムがまだ届きません。いったいどうなって……中略……届きました！ せっかくここまで書いたので，捨てるのももったいないので出すことにしました。どうも，ありがとう。
姫：どうも，お待たせしました。
小羊：バグが多くて困ってます。システムエラーやアドレスエラーが出て，どのプログラムも動きません。
姫：それはバグではありません。ディスクがクラッシュしているのです。すぐにお取り替えします。
小羊：この値段でこの分厚いマニュアルは，史上最強のプロテクトだ！
姫：ごもっともなことです。コピーするより応募したほうが安かったりして。でもそれは実費しかいただいていない証拠ですし，マニュアルを薄くするわけにもいかないし，オンラインマニュアルは見にくいし……どうすればよいのでしょうか？
小羊：私からカンパひとロ。私の母からもうひとロ。母よりひと言：皆さんご苦労様ですが，がんばってください。
姫：あっどうもごていねいに。母上様にもよろしくお伝えください。
小羊：掃除，洗濯なんでもしますから，冬休みに遊びに行っていいですか？
姫：はい，歓迎します。DōGAは年中無休ですが，事前にお手紙で日時をご連絡くださいね。
小羊：友人にCGAシステムを見せたら，「10MHzの68000でこんなスピードが出るわけない。なにか特別なハードを付けたのだろ」といわれた。そうさ，ボクのX68000はポリゴナイザー無い蔵！
姫：おあとがよろしいようで……。

スを作ったときには非常に面倒なことになりかねません。そこでというか，当然というか，mov, rot, scal などのコマンドは，「{」と「}」を用いて，その機能の有効範囲を設定することができます（例6，図7）。

例6を見ながら，mov を例にしたコマンドの有効範囲を解説いたしましょう。この例は，視点に向かってやってくる空飛ぶ「JYUTAN」の上に，スルスルと横滑りする「DESK」と「CHAIR」があり，その「DESK」の上には2冊の「BOOK」があるというわけのわからないものです。

まず，「{」と「}」の対応について見てみましょう（これが非常に重要なのです）。6行目の先頭に付いている「{」は，15行目で閉じています。この中に，7行目から始まって11行目で閉じる括弧と，12行目から始まって14行目で終わる括弧があります。さらに，9行目，10行目は，その1行内で括弧が対応しています。

結論からいうと，各コマンドは括弧の外に対しては影響がなく，その括弧内か，さらに内側の括弧の内部に対してのみ有効となるのです。

6行目の mov の括弧は15行目で閉じているので，この範囲のすべての mov に影響があります。この6行目は「JYUTAN」の動きを記述しているのですから，「JYUTAN」が動けば，その上に乗っているすべての物が動くというわけです。それに対して，8行目の mov は7行目から11行目の括弧に囲まれているので，11行目までしか影響がありません。もちろん9行目，10行目については，さらに内側の括弧なのですから影響があり，「DESK」が動けばその上の「BOOK」は動くというわけです。しかし，「DESK」が動いても，13行目の「CHAIR」は動かず，まったく別の動きをするわけです。

9，10行目は，その行で完結しているので，この行のmovはほかの行に影響を与えません。もし，この「BOOK」を動かしても，ほかの物は動かないわけです。同様に，13

例5. 1　scalの使い方　その1

```
1: #frame ( fno, 1, 20 )
2: fram {
3:      light pal( rgb ( 1 1 1 ) 5 2 -6 )
4:      { mov ( -500 ¥fno*50¥ 200 ) eye deg( 60 ) }
5:      {
6:           mov ( 0 200 0 )       obj CAT
7:           mov ( 0 200 0 )       obj CAT
8:           mov ( 0 200 0 )       obj CAT
9:      }
10: }
11: #endframe
```

例5. 2　scalの使い方　その2

```
1: #frame ( fno, 1, 20 )
2: fram {
3:        light pal( rgb ( 1 1 1 ) 5 2 -6 )
4:        { mov ( -500 ¥fno*50¥ 200 ) eye deg( 60 ) }
5:        {
6:          mov ( 0 200 0 )                 obj CAT
7:          mov ( 0 200 0 )                 obj CAT
8:          mov ( 0 200 0 )  scal ( 2 2 2 )  obj CAT
9:        }
10: }
11: #endframe
```

例5. 3　scalの使い方　その3

```
1: #frame ( fno, 1, 20 )
2: fram {
3:        light pal( rgb ( 1 1 1 ) 5 2 -6 )
4:        { mov ( -500 ¥fno*50¥ 200 ) eye deg( 60 ) }
5:        {
6:                       mov ( 0 200 0 ) obj CAT
7:         scal ( 2 2 2 ) mov ( 0 200 0 ) obj CAT
8:                       mov ( 0 200 0 ) obj CAT
9:        }
10: }
11: #endframe
```

図6　scalの使い方

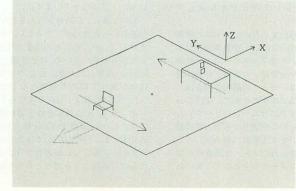
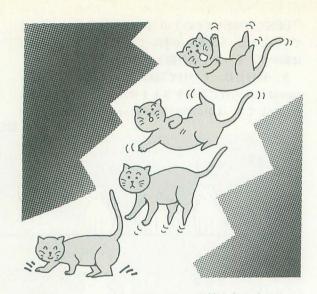
その1

その2

その3

例6　コマンドの有効範囲

```
1:  #frame ( fno, 1, 20 )
2:  fram {
3:       light pal( rgb ( 1 1 1 ) 5 2 -6 )
4:       { mov ( -2000 0 500 )   eye deg(60) }
5:       { mov (     0 0  50 )   target  }
6:       { mov ( ¥div(0,-1500,1,20,fno)¥ 0 100 ) obj JYUTAN
7:         {
8:           mov ( 200 ¥-200+20*fno¥ 0 )      obj DESK
9:                { mov ( 50 50 100 )          obj BOOK }
10:               { mov ( -20 0 100 )          obj BOOK }
11:        }
12:        {
13:          mov ( -200 ¥200-20*fno¥ 0 )      obj CHAIR
14:        }
15:      }
16: }
17: #endframe
```

図7　例6の解説図

行目の mov も「CHAIR」にしか影響を与えません。

この例では mov で解説しましたが，rot や scal でもまったく同じように考えることができます。この「{」と「}」によって，非常に自由度の高い記述が可能となるのです。しかし，Human68kのエディタEDなどでフレームソースを書くと対応を間違えたり，「{」の数と「}」の総数が異なったりすることがよくあります。なにかおかしかったら，まずこの括弧をチェックすることをおすすめします。

おわりに

どうだ，まいったか。今回のフレームソースは難しかったでしょう。1回で終わるつもりだったのですが，次回に引き継ぐことになってしまいました。言っておくけど，来月はもっと難しくなります（オイオイ，死んだフリをしている場合じゃないって）。

今回のフレームソースの mov，rot や「{ }」の使い方は，ほかの本格的なCGシステムにも共通する概念です。最初は戸惑うけど，一度理解できれば簡単，簡単。来月までに，ちゃんとマスターしておいてください。

来月は，いよいよお待ちかねの構造体が出てきます。つまり，「パロレイバー」や「ナイト オブ グリーン」によるアクションデザインを解説するのです。あなたは「パロレイバー」対「ナイト オブ グリーン」のバトルシーンでも作りますか？ それとも“ドジョウすくい”でもやらせますか？

ワ〜イ，ワ〜イ，にこ・にこ・ぷん！

各読者通達事項

[SIG開催のお知らせ]

このたび，DōGAでは，「J&P HOTLINE」において，SIGを開くことになりました。「J&P HOTLINE」は，関西を基点として全国にアクセスポイントを持つ関西最大のパソコン通信です。

・「CGAデータベース」
　形状データなどのライブラリー
・「CGA　Q&A」
　CGAシステムを中心とした，Q&A
・「プログラムライブラリー」
　新しいツールなどのライブラリー（PDS）
・「おしゃべりコーナー」
　ワ〜イ，ワ〜イ，にこ・にこ・ぷん！
・「バグ情報」
　バグなんてだいっきらいだー

といったコーナーを考えており，全国各地のチームとの通信欄としても使用します。一応，年始から開局できる見通しですが，詳しくは来月号の誌面でお知らせします。今後ネット関係の情報は，「遊び人 松井のLogin」のコーナーで扱っていく予定ですので，楽しみにしていてください(シグオペ：遊び人 松井　サブオペ：ペコチャン 上野)。

[DōGAの作品がレンタルビデオショップに！]

先月号でも少し触れましたが，Dō-GAの最新CGA作品「Thank you VOYGER」が集英社ビジネスジャンプ（BJ）の映像フェスティバル「映像大賞」の236本の中から最終ノミネート作品に選ばれました。このコンテストの最大の特長は，最終ノミネート作品10本をまとめて，全国のレンタルビデオショップに「ムービーリーグ」として店頭に並べ，一般の視聴者からの人気投票を行うことです。いままで写真（静止画）でしかご覧いただけなかったCGAを，全国各地の皆さんにちゃんとアニメーションとしてご覧いただける最高のチャンスです（11月16日，23日にはオンエアされたけど誰も見ていなかっただろうな)。最初からそれが目的で出品したのですが，第1回目のコンテストにしてはハイレベルな作品が多数集まっていて，一時は入賞をあきらめていました。結局は，やはりCGというものの珍しさが有利に働いたのではないでしょうか。ほかの9作品は，セミプロが制作したものや制作費150万円という本格派が揃っています。“映像”の勉強のためにもぜひ一度ご覧ください。期間は1月中旬まで，ビクター系列のレンタルビデオショップには必ず置いてあるそうです。

すでに，この作品をご覧になった方からのお手紙が何通か来ています。本当にDōGA・CGAシステムで作ったのか疑っていらっしゃるようですが，それではほかのどのシステムを用いれば可能なのでしょう。当チームがGWS(グラフィック・ワークステーション）を持っているわけないし，市販のレイトレソフトを使ったとしたら，計算時間だけで数年間必要となるでしょう。ただし，静止画の一部やマッピングデータにZ's STAFFを，レンダリングはRENDを32ビットパソコン（Panacom M800）に移植して高速化するといったことは行っています。

しかし逆に，CGAシステムを用いれば誰にでもああいった作品ができるというわけではありません（いや，私はあの程度の作品なら誰にでもできると思っている。自己評価：100点満点で40点)。長年にわたる，血と汗のつらい修行の日々があって初めて可能となるのです（大ウソ)。“CGAの道は1日にしてならずじゃ”

ところで，BJに掲載されていた批評は，「発想は並。しかしCGの技術は圧巻！と票が集結」とありました。「発想は並」とはえらい言われようですが，ご覧になればおわかりのように，あの作品は“奇抜さ”を狙ったものではなく，意図的にテーマも演出も表現もオーソドックスに（あるいはやや古典的に）まとめたのですから，的のはずれた批評と思うのですが……。

[CGAシステム発送先不明]

以下の方々は，指定された住所に発送しましたが，あて先不明で戻ってきております。正確で詳しい住所，氏名，電話番号を，ていねいに明記してご連絡ください。

愛知県豊田市花園町	加藤　様
神奈川県横浜市西区	宮澤　様
三重県四日市市河原田町	石？　様
群馬県豊岡市豊岡	平木　様
滋賀県大津市長等	山田　様
愛知県西春日井郡豊山町	岩浪　様

「電信振込」で申し込まれた方については，基本的に対応いたしかねます。“そんなむごい……”なんて言わせません。なぜなら「電信振込」には，振込人の住所も名前も書かれていないのです。発送できるわけないじゃないか！

以下同様に，発送できるわけない方々の特集です。心当たりのある方は，正確な住所，氏名，電話番号を明記して，振り込んだ本人である証拠のコピーを同封して，ご連絡ください（あんまり世話を焼かせるんじゃないよ)。

住所不明	シシクラ　ヒカル　様
住所不明	ヒシヌマ　ユウジ　様
住所不明	イトウ　ツトム　様
住所不明	ツジ　マサトシ　様
住所不明	モリ　カズヒデ　様
住所不明	山崎　勝之助　様
群馬県伊勢崎市八斗島町	？様
6月29日に大阪府守口市から振り込んだ	？様
8月10日に藤井寺市から振り込んだ	？様
9月28日に東京都品川区から振り込んだ	？様
10月24日に振り込んだ	？様
10月30日に振り込んだ	？様

X1/X1turbo用
さよならを過ぎて
X68000用
RYDEEN

Fushiki Yoshihiro
伏喜 義宏

Ageno Yuji
揚野 雄二

あけましておめでとうございます。このページもLIVE in '90になり，心機一転がんばりたいと思います。そこで，「たまにはゲーム・ミュージック以外も載せてほしい」という声に応えて今月はポピュラーを2曲用意しました。こたつに入って存分にお楽しみください。

いきなりごめんなさいのコーナーです。'89年12月号でX1用の天空の城ラピュタより「パズーとシータ」という名曲をお届けしたはずでした。ところが，このリストには「音色のDATAが抜けている」というご指摘が山のようにありました。確かにそのとおりです。すべて私がやったことです。ごめんなさい。まず，リスト1を入力して，先に実行させるようにしてから，12月号のDATAを演奏させるようにしてください。サブルーチンにしてしまうのが楽でしょう。具体的には，リスト1を12月号のリストの2000行から入力して，最後にRETURNを追加します。それから，85行にGOSUB2000を入れてください。

さて，気を取り直して今月の作品の紹介にいきましょう。今月もX1MusicBASIC用とX68000OPMA用です。最近めっきりとMZ-2500の投稿が少なくなりました。確かに，Xシリーズに比べて音源が弱いというのはわかりますが，FM音源3声＋PSGでもちゃんと音楽できるはずです。君はPC-8801mkIISRのテグザーのショックを忘れたのか？ 25ユーザーがんばれ！

マンモスうれP，初登場

完全に常連と化してしまった，伏喜君の登場です。あるときはT-SQUARE，またあるときは聖飢魔II，しかしてその実体はっっっ！ のりピーマ～ン。……失礼しました。あまりにも伏喜君のレパートリーが広いので，ついやってしまいました。

「いただきマンモス」「ごちそうサマンサ」など，のりピー語で有名なアイドル，酒井法子さんの登場です。曲は「さよならを過ぎて」。恥ずかしながら私は原曲を聴いたことがありませんが，バランスのよさはさすがに常連といったところでしょうか。原曲とFM音源とのマッチングがよいのか，アレンジがうまいのかなかなか曲選びのセンスも素晴らしいみたいですね。これからもがんばってください。私としては一部ドラムで気にかかるところ（タムなのだと思いますが，ポ～ンって音）がありましたが，ほかでカバーしてるので良しとしましょう。気になった人は，音程を下げてみてください。プログラムは，リスト2をRUNすると，リスト3を読みにいきます。ひっさしぶりのアイドル路線を堪能してください。

YMOは不滅です！

X68000OPMA用には，かの有名なYMOの「RYDEEN」です。パソコン業界では，この曲のないMMLはモグリだ！ と言われるほどのスーパー・メジャー・ソングです。知らなかった人は，ぜひ打ち込んで聴いてみてください。ひょっとしたら，友達のコンピュータとかで聴いたことがあるかもしれません。なにをいまさら，と思ったあなたはまだ甘い。曲のデキのほうは，3重丸をあげたくなるぐらい気合が入っていて，私が聴いたところでは，パッパッカッ・パッパッカ……という馬のひづめ？ 以外はほとんど再現されています。ここまでくるとFM音源の性能のレベルの話までいってしまいますね。正直言ってみごとなもんです。

まずは，リスト4を打ち込んで聴いてみてください。もし言いたいことがあったら，その後にどうぞ。以前に「こだわり」の話をしましたが，この曲からはその「こだわり」がひしひしと伝わってきます。マニアをも唸らせるデキ，さてはよほどの常連か？ と思いきや，これが初投稿というのだから，世の中広いもんだ。しかも，X68000上では自作第1号のプログラムというのだから素晴らしい。久々に期待の大型新人登場ということで，常連さんもうかうかしてはいられなくなりました。今年のOh!X LIVEを面白くするのは，あなたかもしれません。あなたの熱意のこもった作品，お待ちしております。 (S.K.)

酒井法子

リスト1 パズーとシータ音色データ

```
10 '
20 '     LAPUTA  Pazu & Sheeta   TONE DATA
30 '
40 MEM$(&HB1D8,36)=HEXCHR$("FC 00 31 52 50 30 1A 06 16 06 1F 11
1F 11 1F 1F 1F 1F 00 00 00 00 02 06 02 06 00 00 00 00 00 C8 80 0
0 02 00")          'I3
50 MEM$(&HB1FC,36)=HEXCHR$("FA 00 32 14 76 04 1B 1E 24 00 5D 5D
5D 4A 1F 1F 1F 1F 00 00 00 00 05 05 04 06 00 00 00 00 F6 C8 80 0
0 02 00")          'I4
60 MEM$(&HB244,36)=HEXCHR$("FA 11 51 25 71 11 25 3E 4D 00 5D 56
5D 9F 05 00 00 87 07 04 04 06 94 45 45 45 00 80 80 00 00 DC 80 0
4 02 80")          'I6
```

MusicBASICは1988年12月号で発表されたX1用MML，OPMAは1989年4月号で発表されたFM音源とPCM音源を同期させるX68000用ミュージックドライバです（OPMAがなくてもFM音源の演奏は可能です）。

日本音楽著作権協会(出)許諾第8972064-901号

リスト2 さよならを過ぎて・1

```
10 '
20 '      「 ｻﾖﾅﾗｦ ｽｷﾞﾃ 」     by NORIKO SAKAI
30 '
40 '        Tone Set Program  by Y.FUSHIKI
50 '
60 MEM$(&HB190,36)=HEXCHR$("EA 40 36 74 71 31 30 23 2A 0A 13 15
0B 0D 12 0E 1F 9F C4 C6 03 01 54 6A 0A 08 80 80 80 00 0C C6 B2 1
4 02 02")      ' 1 NORI-P
70 MEM$(&HB1B4,36)=HEXCHR$("EC 40 35 76 73 31 1D 00 2A 0A 13 15
0B 14 00 09 00 89 04 03 03 03 04 65 05 67 80 80 80 00 0C C6 8A 0
0 02 01")      ' 2 ｲﾝﾄﾛ
80 MEM$(&HB1D8,36)=HEXCHR$("FA 00 51 25 71 11 25 39 43 00 1F 1D
1D 1F 05 00 00 88 07 04 04 06 94 55 55 F5 00 80 80 00 00 00 80 0
0 00 00")      ' 3 A PIANO
90 MEM$(&HB1FC,36)=HEXCHR$("DC 00 3E 02 33 31 2F 00 21 1B 1F 1B
1E 1E 0A 88 06 86 C0 04 0B 0B 5A B6 F6 F6 00 00 00 00 00 C8 80 0
0 02 00")      ' 4 Synth1
100 MEM$(&HB220,36)=HEXCHR$("FA 00 51 25 71 11 25 39 43 00 1F 1D
 1D 1F 05 00 00 8B 07 04 04 06 94 55 55 F5 00 80 80 00 00 00 80
00 00 00")      ' 5 String1
110 MEM$(&HB244,36)=HEXCHR$("FC 00 41 42 42 42 1E 2F 1B 00 1F 1F
 19 10 00 09 14 85 00 00 00 00 59 A6 75 AB 00 00 00 00 00 00 80
00 02 00")      ' 6 String2
120 MEM$(&HB268,36)=HEXCHR$("FA 00 36 7F 73 32 2F 1B 39 00 1F 1F
 14 17 0D 12 0A 85 01 01 01 01 F4 FA F7 F5 00 80 00 00 00 D2 80
00 02 00")      ' 7 String3
130 MEM$(&HB28C,36)=HEXCHR$("FA 00 30 7D 73 31 25 39 25 1E 1F 1F
 14 17 0D 14 0A 8C 01 81 01 01 F4 FA F7 F7 00 80 00 00 00 D2 80
00 02 00")      ' 8 W BASS
140 MEM$(&HB2B0,36)=HEXCHR$("C4 00 53 04 53 01 1B 1B 0C 00 1F 1F
 1F 1F 04 88 04 88 00 01 00 01 02 F8 02 F8 00 00 00 00 00 C8 80
00 02 00")      ' 9 Cembalo
150 MEM$(&HB2D4,36)=HEXCHR$("F8 00 35 30 30 31 2F 25 19 13 1E 1C
 1C 1C 0D 06 04 81 08 0A 03 05 B6 B6 36 26 00 00 00 00 00 C6 80
00 02 00")      '10 BASS
160 MEM$(&HB2F8,36)=HEXCHR$("FA 00 01 75 01 41 23 20 2F 0C 14 19
 1C 0C 02 06 03 84 00 00 00 00 15 38 16 06 00 00 00 00 F4 00 80
00 00 00")      '11 Violin
170 MEM$(&HB31C,36)=HEXCHR$("FB 00 70 52 20 30 1D 2A 23 00 1B 1F
 1F 1D 12 0A 0A 0C 01 01 01 01 F4 F3 F3 F6 00 00 00 00 00 96 80
00 02 00")      '12 W BASS2
180 MEM$(&HB340,36)=HEXCHR$("BA 00 43 23 71 41 2A 1B 2F 00 13 13
 13 13 0C 0C 02 8A 01 01 01 01 54 54 34 57 00 00 00 00 00 CD 80
00 02 00")      '13 A PIANO2
190 MEM$(&HB388,36)=HEXCHR$("FA 50 01 70 00 41 23 25 2F 0C 0F 0F
 14 04 02 06 03 84 00 00 00 00 15 38 16 06 00 00 00 00 F4 CA 8A
03 02 05")      '15 Violin 2
200 MEM$(&HB3D0,36)=HEXCHR$("F9 00 04 02 04 01 04 0F 00 00 1A 1C
 1C 1A 0A 12 19 92 00 40 00 00 AA FA FA F7 00 00 00 00 F4 C8 80
00 02 00")      '17 ﾊﾟｰｶｯｼｮﾝ
210 MEM$(&HB3F4,36)=HEXCHR$("F2 00 00 01 31 01 15 17 25 00 18 1A
 1F 1A 14 0A 0A 89 00 80 40 00 FA F6 F4 F5 00 00 00 00 F4 8C 80
00 02 80")      '18 Tom-Tom
220 MEM$(&HB5C8,36)=HEXCHR$("FA 00 31 22 41 61 11 2F 39 14 10 0F
 0D 1F 0C 0C 0A 8A 01 01 00 00 A7 A7 17 07 00 00 00 00 00 C8 80
00 02 00")      '31 Trmpet
230 MEM$(&HB5EC,36)=HEXCHR$("FA 00 0E 00 05 01 00 00 00 2F 1F 1F
 1F 10 00 00 00 93 00 00 00 00 02 02 02 F9 00 00 00 00 18 C8 80
00 02 00")      '32 Shaker
240 MEM$(&HB658,36)=HEXCHR$("FB 00 71 06 03 3D 12 0D 16 00 1E 1E
 1F 12 04 01 14 8A 40 00 00 00 F1 F1 F1 F5 80 00 00 00 0D C8 80
00 02 00")      '35 Ride Cymb
250 MEM$(&HB67C,36)=HEXCHR$("FB 00 0E 08 07 00 0F 07 11 05 1A 1A
 1A 16 04 08 0C 11 40 00 80 00 32 72 BA F8 00 00 00 00 00 C8 80
00 03 00")      '36 H-H Close
260 MEM$(&HB6C4,36)=HEXCHR$("FE 00 08 31 40 00 00 04 04 04 1F 1F
 1F 1F 04 11 0F 0A 40 10 0D 0D 24 97 89 89 00 00 00 00 FD C8 80
00 03 00")      '38 Tom
270 MEM$(&HB6E8,36)=HEXCHR$("FC 00 0E 00 00 00 00 00 00 01 1F 1F
 1F 1F 00 10 16 10 C0 CE 54 0E 05 67 55 67 00 00 00 00 F4 00 80
00 03 00")      '39 Snare Drum
280 MEM$(&HB70C,36)=HEXCHR$("83 00 08 0F 00 00 01 00 07 00 1E 1E
 1E 1E 1A 1C 0F 0F 00 DF 00 00 FD AE F8 F8 00 00 00 00 00 C8 80
00 02 00")      '40 BASS Drum
290 '
300 RUN"ｻﾖﾅﾗｦｽｷﾞﾃ 2.mml"
```

リスト3 さよならを過ぎて・2

```
10 '
20 '      「 ｻﾖﾅﾗｦ ｽｷﾞﾃ 」     by NORIKO SAKAI
30 '
40 ' Programed by Y.FUSHIKI 1989/9/1-1989/10/6
50 '
60 'SAVE "ｻﾖﾅﾗｦｽｷﾞﾃ 2  .mml"
70 '
80 INIT : CLS0 : DEFINT A-Z : DEFSTR A-H,T : DIM A(31)
90 PLAY0 : PLAY"T127"
100 LOOP=0
110 T1="C.>E.<C16C16C.>E.<C16C16
120 T2="C.>E.<C16C16C16>E16<C16>E.<C16C16
130 GOTO 220
140 LABEL "!"                              :'Drum Sub
150 K=INSTR(A(J),"I40") : IF K=0 THEN 180
160 A(J)=LEFT$(A(J),K-1)+"i40o0v15P3"+MID$(A(J),K+3,LEN(A(J))-K)
170 GOTO 150
180 K=INSTR(A(J),"I39") : IF K=0 THEN 210
190 A(J)=LEFT$(A(J),K-1)+"i39o2v16P3"+MID$(A(J),K+3,LEN(A(J))-K)
200 GOTO 180
210 RETURN
220 '
230 'MAIN
240 A(0)="L8I2O5V10Q8P2K0S2,2,0,10=1
250 A(1)=STRING$(2,"F4EF&FG4F& F1")
260 A(2)="F4EF&FG4F& F2G2 A2&A4.A& A2&P3A2
270 A(3)="=0I3O5V14A.A16&AB-A4&AD16F16 A4G4>C4<B-4 G.G16&GAG4&GC
16E16
280 A(4)="G4F4>C4&C<F16G16
290 A(5)=A(3)
300 A(6)="G4F4>C4I1O5V14=0DE
310 A(7)="I1O5F4FEFGED16C16& C2R4<A>C
320 A(8)="D4DFEDCD <A2R4RA>
330 A(9)="<B-4>A4G4RD C4G4F4EF
340 A(10)="G.G16&GFEDFE& E2R4DE
350 A(11)="F4FEFGED16C16& C2R4<A>C
360 A(12)="D4DFEDCD <A2R4RA
370 A(13)="B-4>A4G4RD C4B-4A4GA
380 A(14)="B-F4DEF4E D4R4D4E4
390 A(15)="F4.<B-B-2> R4G4F4E4
400 A(16)="G4.CC2 R4A4F4E4
410 A(17)="F.G16&GD&D4DE F.G16&GD&D4RE
420 A(18)="F.G16&GAA.G16&GF A1
430 A(19)="R2F4G4
440 A(20)="A.A16&AB-A4FG A.A16&AB-A4RG
450 A(21)="A4GFEFGA& A2R4RA
460 A(22)="GF+GA16A16G4RG FEFG16G16F4RD
470 A(23)="E.E16&EDG.G16&GF A2F4G4
480 A(24)=A(20)
490 A(25)=A(21)
500 A(26)="GF+GA16A16G4RG FEFG16G16F4DE
510 A(27)="F.E16&ED&D2 R4EFG.F16&FE
520 A(28)="E.D16&DD&D2 R1
530 A(29)="R1 R2.DE
540 FOR J=0 TO 29:PLAYA(J);:NEXT
550 FOR J=7 TO 27:PLAYA(J);:NEXT
560 PLAY "E.D16&DD&D2 I13O6V9R8R4G4A4B-4";
570 PLAY ">C1< R4G4F4.E";
580 PLAY "F1 E4F4G4&GE16.D32&";
590 PLAY "E16&E16&E2.E16E16 D4.I1O5V14D4E4";
600 FOR J=15 TO 27:PLAYA(J);:NEXT
610 FOR I=0 TO 1
620 IF I=0 THEN PLAY "E.D16&DD&D2 I3O5V12R16A4G4>C4<B-4";:GOTO 6
40
630 PLAY "A.A16&AB-A4&AD16F16  A4G4>C4<B-4";        :'Fade out
640 PLAY "G.G16&GAG4&GC16E16 G4F4>C4<B-4";
650 PLAY "A.A16&AB-A4&AD16F16 A4G4>C4<B-4";
660 PLAY "G.G16&GAG2 R1";
670 NEXT
680 PLAY":";
690 '
700 'ECHO
710 A(0)="L8I2O5V8 Q8P3K0S2,2,0,10=1R8
720 A(3)="=0I3O5V12P3A A16&AB-A4&AD16F16 A4G4>C4<B-4 G.G16&GAG4&
GC16E16
730 A(4)="G4F4>C4&C<F16
740 A(5)="I15O6P3V9=1 P3A1& A1 G1&
750 A(6)="G2. I1O5V9K5R8=0 DE
760 A(19)="R4.D4E4
770 A(20)="F.F16&FGF4DG F.F16&FGF4RRG
780 A(23)="E.E16&EDG.G16&GF A4.D4E4
790 A(24)=A(20)
800 A(28)="E.D16&DD&D4I6O5V9=1P1A4 F1
810 A(29)="R2.G4 C2.I1O5=0V9P3DE
820 FOR J=0 TO 29:PLAYA(J);:NEXT
830 FOR J=7 TO 27:PLAYA(J);:NEXT
840 PLAY "E.D16&DD&D2 R1";
850 PLAY "R1 R1";
860 PLAY "R1 R1";
870 PLAY "R1 R2D4E4";
880 FOR J=15 TO 26:PLAYA(J);:NEXT
890 PLAY "F.E16&ED&D2 R4EFG.F16&FI3O5V14";
900 FOR I=0 TO 1
910 PLAY "A.A16&AB-A4&AD16F16 A4G4>C4<B-4";
920 PLAY "G.G16&GAG4&GC16E16 G4F4>C4<B-4";
930 PLAY "A.A16&AB-A4&AD16F16 A4G4>C4<B-4";
940 PLAY "G.G16&GAG2 R1";
950 NEXT
960 PLAY":";
970 '
980 'Synth1
990 A(0)="L8I2O5V5 Q8P3S2,2,0,10=1
1000 A(1)=STRING$(2,"D4CD&DE4D& D1")
1010 A(2)="D4CD&DE4D& D2E2 F2&F4.F& F1
1020 A(3)="=0I3O5V12F.F16&FGF4&F<B-16>D16 F4E4A4G4 E.E16&EFE4&E<
A16>C16
1030 A(4)="E4D4A4&AD16E16
1040 A(5)=A(3)
1050 A(6)="E4D4A4R4
1060 A(7)="I5O5P3V10D1 A1
1070 A(8)="B-2C2 I4O5V12=1E4.C4<A4.>=0
1080 A(9)="I5O5V10G2C2 A2D2
1090 A(10)="E1 I4O5V12=1R16AA16A8.AA16A4&A=0
1100 A(11)="I6O4P1V12D1 A1
1110 A(12)="B-2C2 I4O5P3V12=1E.D.C.<A.G4>=0
1120 A(13)="I6O4P1V12G2C2 A2D2
1130 A(14)="G2A2 D1
1140 A(15)="I7O6P2V7=1G32A32B-.&B-2. A32B-32>C.&C2.
1150 A(16)="<F32G32A.&A2. B-32>C32D.&D2.=0
1160 A(17)="I9O4K0P3V8G.G16&GG&G4R4 G.G16&GG&G4R4
1170 A(18)="G.G16&GGG.G16&GG V11{AAA}2{AAA}2
1180 A(19)="AI18O2V13A16A16A4R2
1190 A(20)="I3O5V11F.F16&FGF4R4 F.F16&FGF4R4
1200 A(21)="I9O4=3V10R1 R8A16A16A16A16R16A16>C16R16<B-16R16A4=0
```


▶私はかつてキャベツのしぼり汁のカンジュースを飲んだことがあります。それもまじりっけなしの天然果汁(?)100%! もちろんひと口で捨てました。実はそれは友人の姉の勤め先で冗談で作ったものだったらしいのですが。こんな幻の毒物(?)飲料を飲んだ経験のある人は，そういないでしょう。 中村 幸司(26)東京都

```
1210 A(22)="R1 R1
1220 A(23)="R1 I3O4V11L16GAB->CDEFGA2L8
1230 A(24)=A(20)
1240 A(25)=A(21)
1250 A(26)=A(22)
1260 A(27)="I4V12O4P3F1 E1
1270 A(28)="I3O4V10D1 I13O5V7=1P2A16F16D&D2.
1280 A(29)="B-16G16E&E2. G16E16C&C2.=0
1290 FOR J=0 TO 29:PLAYA(J);:NEXT
1300 FOR J=7 TO 27:PLAYA(J);:NEXT
1310 PLAY "I13O6V12A1 R4G4A4B-4";
1320 PLAY ">C1< R4G4F4.E";
1330 PLAY "F1 E4F4G4&GF16.D32&";
1340 PLAY "E32&D16.&D2.D16E16 D1";
1350 FOR J=15 TO 27:PLAYA(J);:NEXT
1360 FOR I=0 TO 1
1370 PLAY "I3O5V12F.F16&FGF4&F<B-16>D16 F4E4A4G4";
1380 PLAY "E.E16&EFE4&E<A16>C16 E4D4A4G4";
1390 PLAY "F.F16&FGF4&F<B-16>D16 F4E4A4G4";
1400 PLAY "E.E16&EFE2 R1";
1410 NEXT
1420 PLAY":";
1430 '
1440 'Synth2
1450 A(0)="L8I1O5V12Q8P3S4,5,130,0=3
1460 A(1)=STRING$(2,"F4EF&FG4F& F1")
1470 A(2)="F4EF&FG4F& F2G2 A2&A4.A& A1S2,2,0,10
1480 A(3)="=0I31O4L16R2R8V12P2AP1AV10P2AP1AV8P2AP1A R1 R2R8V12P2
GP1GV10P2GP1GV8P2GP1GL8
1490 A(4)="R1
1500 A(5)=A(3)
1510 A(6)=A(4)
1520 A(7)="K6"+A(7)
1530 A(8)="B-2C2 I4O5V7=1C4.<A4F4.>=0
1540 A(10)="E1 I4O5V12=1R16AA16G8.FF16E4&E=0
1550 A(11)="I6O4P2V12D1 A1
1560 A(12)="B-2C2 I4O5P3V12=1E.D.C.<A.G4>=0
1570 A(13)="I6O4P2V12G2C2 A2D2
1580 A(14)="G2A2 D1
1590 A(15)="I7O6P1V7=1R16G32A32B-&B-2. R16A32B-32>C&C2.
1600 A(16)="<R16F32G32A&A2. R16B-32>C32D&D2.
1610 A(18)="G.G16&GGG.G16&GG V7{EEE}2{EEE}2
1620 A(19)="EI18O1V11A16A16A4R2
1630 A(20)="I11O6V8=1D1& D1&
1640 A(21)="D1 C1
1650 A(22)="<B-2A2& A2.>F4
1660 A(23)="D1 D-1
1670 A(24)=A(20)
1680 A(25)=A(21)
1690 A(26)="<B-2A2& A1
1700 A(27)="I4V12O4P3D1 C1
1710 A(28)="I3O3V10A1 I13O5V5P1=1F16D16<B-&B-2.
1720 A(29)=">G16E16C&C2. E16C16<A&A2.=0
1730 FOR J=0 TO 29:PLAYA(J);:NEXT
1740 FOR J=7 TO 27:PLAYA(J);:NEXT
1750 PLAY "I13O6P3V9RA2.&A I6V7O5C1";
1760 PLAY "<A1> D1";
1770 PLAY "G1 <A1>";
1780 PLAY "I9O5V5D.DR16DR16DD16D4 D.DR16DR16DD16D4";
1790 FOR J=15 TO 27:PLAYA(J);:NEXT
1800 FOR I=0 TO 3
1810 PLAY A(3)+"R1";
1820 NEXT
1830 PLAY":";
1840 '
1850 'BASS
1860 A(0)="L8Q8I12Q3V15P3
1870 A(1)="D1& D2D4C4 <B-1& B-2B-4A4
1880 A(2)="G1& G2R>GD<G A2R>AE<A& A1
1890 A(3)="I10V11O3D.DR16EF4D4 <B-.B-R16>CD4<B-4> C.CR16DE4C4
1900 A(4)="<A.AR16GA4>C4
1910 A(5)=A(3)
1920 A(6)=A(4)
1930 A(7)="V11D.DR16CD4D4 <A.AR16GA4>C4
1940 A(8)="<B-.F.B->C4<G4 A.AR16B->C4<A4
1950 A(9)="B-.B-.>DC4<G4 A.AR16>ED4CD
1960 A(10)="E.ER16E16R16E.&E-64&D64&D-64&C64&<B-1 A2.&>A1
1970 A(11)="D.DR16CD4D4 <A.AR16GA4>C4
1980 A(12)="<B-.F.B->C4<G4 A.AR16B->C4<A4
1990 A(13)="B-.B-R16>DC4<G4 A.AR16>ED4CD
2000 A(14)="<G.>D.<GA.>E.<A >D2S4,5,0,7=1D4C4=0
2010 A(15)="<B-1 B-1
2020 A(16)="A1 >D1
2030 A(17)="<GR16GR16G4.&G.>!110G16 FR16FR16F4.&F4
2040 A(18)="E.ER16EE.ER16E <{AAA}2{AAA}2
2050 A(19)="A4R2S4,5,0,15=2A4=0
2060 A(20)=">D.DR16EF4D4 C.CR16DE4C4
2070 A(21)="<B-.B-R16B->C4<G>G <A.AR16B->C16R16<B-A4
2080 A(22)="G.GR16GA4>D-4 D.<A.>DC4<A&>C
2090 A(23)="<B-.B-R16B-2G A.AR16GA4A4
2100 A(24)=A(20)
2110 A(25)=A(21)
2120 A(26)=A(22)
2130 A(27)="<B-.B-R16GB-.B-B-16B- A1>
2140 A(28)="V10D1 V9<B-1
2150 A(29)=">C1< A1>
2160 FOR J=0 TO 29:PLAYA(J);:NEXT
2170 FOR J=7 TO 27:PLAYA(J);:NEXT
2180 PLAY "<B-.B-R16>CD4<B-4 B-.&B-.B-&>G64&A-64&A16.&A8<B-1";
2190 PLAY "A.AR16B->C4<A4 >D.&D.D&D4C4";
2200 PLAY "<G.GR16AB-4G4 A.&A.B->C4<A4";
2210 PLAY ">D1 D1";
2220 FOR J=15 TO 27:PLAYA(J);:NEXT
2230 FOR J=3 TO 6:PLAYA(J);:NEXT
2240 FOR J=3 TO 6:PLAYA(J);:NEXT
2250 PLAY":";
2260 '
2270 'ﾊﾟｰｶｯｼｮﾝ
2280 A(0)="L8Q8I8 O4V10P3
2290 A(1)=A(1)
2300 A(2)="G1& G2R>GD<G A2R>AE<A& A1 I17O3P1V11
2310 A(3)=T1+T2+T1
2320 A(4)=T2
2330 A(5)=T1+T2+T1
2340 A(6)=T2
2350 FOR I=7 TO 17
2360 A(I)=T1+T2
2370 NEXT
2380 A(18)=T1+"I35P3O7V8{DDR}2{DDR}2
2390 A(19)="!110D2R2I17O3P1V11
2400 FOR I=20 TO 26
2410 A(I)=T1+T2
2420 NEXT
2430 A(27)="C.>E.<C16C16R2 I35O6P3!125D2D2I17O3V11P1
2440 A(28)="I35O6P3!125D2I6P3V12O5=1R4A4 F1
2450 A(29)="R2.G4 C1I17O3V11P1=0
2460 FOR J=0 TO 29:PLAYA(J);:NEXT
2470 FOR J=7 TO 27:PLAYA(J);:NEXT
2480 FOR I=1 TO 4
2490 PLAY T1+T2;
2500 NEXT
2510 FOR J=15 TO 27:PLAYA(J);:NEXT
2520 FOR J=3 TO 6:PLAYA(J);:NEXT
2530 FOR J=3 TO 6:PLAYA(J);:NEXT
2540 PLAY":";
2550 '
2560 'HiHat
2570 PLAY "L8Q7I36O6V10P3S2,2,0,10=1";
2580 PLAY "I2O5V12Q8P1K3"+STRING$(2,"F4EF&FG4F& F1");
2590 PLAY "F4EF&FG4F& F2G2 A2&A4.A& A2=0I39O2P3K0Q7V16EE16E16EE1
6E16";
2600 PLAY "I35O6V15D2I36L16"+STRING$(5,"V11G32G32GGGV8G32G32GGG"
);
2610 PLAY STRING$(4,"G32G32GGG");
2620 PLAY STRING$(6,"V11G32G32GGGV8G32G32GGG");
2630 PLAY STRING$(4,"G32G32GGG");
2640 '
2650 PLAY "V14I35P3D4V9L8"+STRING$(7,"P3I36G64P2I32G16.&G64P3I36
G");
2660 FOR I=8 TO 14
2670 PLAY STRING$(8,"P3I36G64P2I32G16.&G64P3I36G");
2680 NEXT
2690 FOR I=1 TO 3
2700 PLAY STRING$(3,"P3I36G64P2I32G16.&G64P3I36G")+"I39O2P2V16E4
V9O6"+STRING$(3,"P3I36G64P2I32G16.&G64P3I36G")+"I39O2P1V16E4V9O6
I36";
2710 NEXT
2720 PLAY STRING$(8,"P3I36G64P2I32G16.&G64P3I36G");
2730 PLAY "R.I18O2V10A16A16A4R.I35O6V14P1D4P2";
2740 PLAY "I35O6V14D4I36O6L16P3"+STRING$(3,"V8G32G32GGGV11G32G32
GGG")+"V8G32G32GGG";
2750 PLAY STRING$(4,"V11G32G32GGGV8G32G32GGG");
2760 PLAY STRING$(4,"V11G32G32GGGV8G32G32GGG");
2770 PLAY STRING$(4,"V11G32G32GGGV8G32G32GGG");
2780 PLAY "I35O6V14D4I36O6L16P3"+STRING$(3,"V8G32G32GGGV11G32G32
GGG")+"V8G32G32GGG";
2790 PLAY STRING$(4,"V11G32G32GGGV8G32G32GGG");
2800 PLAY STRING$(4,"V11G32G32GGGV8G32G32GGG");
2810 PLAY "V11G32G32GGGV8G32G32GGGR2 R2L8V9P3I36G64P2I32G16.&G64
P3I36G4.";
2820 IF LOOP=1 THEN 2870
2830 IF LOOP=2 THEN 2910
2840 PLAY "V13"+STRING$(8,"I32P2GI36P3G");
2850 PLAY "I32P2GI36P3GI32P2GI36P3GR2 L16I40O0V16P3E4E4E4E4I36O6
V11";
2860 LOOP=1:GOTO 2650
2870 FOR I=1 TO 4
2880 PLAY STRING$(8,"P3I36G64P2I32G16.&G64P3I36G");
2890 NEXT
2900 LOOP=2:GOTO 2690
2910 FOR I=1 TO 2
2920 PLAY "I35O6V15D2I36L16"+STRING$(5,"V11G32G32GGGV8G32G32GGG"
);
2930 PLAY STRING$(4,"G32G32GGG");
2940 PLAY STRING$(6,"V11G32G32GGGV8G32G32GGG");
2950 PLAY STRING$(4,"G32G32GGG");
2960 NEXT
2970 PLAY":";
2980 '
2990 'Bass Drum&SnarDrum
3000 A(0)="L8Q7P3S4,1,0,253
3010 A(1)="Q8I12O3V10P3R16 D1& D2D4C4 <B-1& B-2B-4A4
3020 A(2)="G1& G2R>GD<G16 I40Q7E4E4E4E4 E4E4E4E4
3030 A(3)=STRING$(3,"I40E.I39E16I40E4E4I39E4")
3040 A(4)="I40E.I39E16I40E4E4I39E4
3050 A(5)=A(3)
3060 A(6)="I40E.I39E16I40E4E16I39E16E8I40E16I39E16I40E16I39E16
3070 A(7)=STRING$(2,"I40E4E4E4I39E4")
3080 A(8)=A(7)
3090 A(9)=A(7)
3100 A(10)="I40E4E4E4I39E4 I40E4E4E4I39E16E16E
3110 A(11)=A(7)
3120 A(12)=A(7)
3130 A(13)=A(7)
3140 A(14)=A(7)
3150 A(15)="I40E4E4E4I38O4V15=3P2E4=0 I40E4E4E4I38O4V15=3P1C4=0
3160 A(16)=A(15)
3170 A(17)=A(15)
3180 A(18)="I40E4E4E4I38O4V15=3P2E4=0 I40{EEE}2{EEE}2
3190 A(19)="EI38O4V15P2=3E16E16P1D4=0R4I40P3E4
3200 A(20)=STRING$(2,"I40E.I39E16I40E4E4I39E4")
3210 A(21)=A(20)
3220 A(22)=A(20)
3230 A(23)="I40E.I39E16I40E4E4I39E4 I40E.L16I39EI40E4EI39EE8I40E
I39EI40EI39EL8
3240 A(24)=A(20)
3250 A(25)=A(20)
3260 A(26)=A(20)
```

▶ X68000を購入するため，必死でお金を貯めている私であります。そんな私に今すぐローンを組んでX68000を買いなさい！　といわんばかりのこの記事，この特集。ああっ，Oh！Xよ，あなたが憎い！　これからも充実した記事，特集の数々を載せてください。

後迫　浩一(28)東京都

```
3270 A(27)="I40E.I39E16I40E4I38O4V16=3P2E.E16P3<D16D16P1C=0 R2.R
I39E
3280 A(28)="I40E4E4E4E4 E4E4E4E4
3290 A(29)="E4E4E4E4 I39E16E16E16E.E16E16E16E.EE16E16
3300 FOR J=0 TO 29:"!":PLAYA(J);:NEXT
3310 FOR J=7 TO 26:PLAYA(J);:NEXT
3320 A(27)="I40E.I39E16I40E4RE16I39E16I40E16I39E16E8& E2E32E16.I
40E16I39E16I40E16I39E16E8
3330 A(28)=A(20)
3340 A(29)="I40E.I39E16I40E4E4I39E4 I40E.I39E16I40E4E4I39EE16E16
3350 A(30)=A(20)
3360 A(31)="I40E.I39E16I40E4E4I39E4 I40E.I39E16I40E4E4I39EP2I38O
4V16=3P2E=0
3370 FOR J=27 TO 31:"!":PLAYA(J);:NEXT
3380 FOR J=15 TO 26:PLAYA(J);:NEXT
3390 A(0)="I40E.I39E16I40E4EI40E16I39E16I40E16I39E16E8 R1
3400 J=0:"!":PLAYA(J);
3410 FOR I=0 TO 1
3420 PLAY A(20);
3430 A(0)="I40E.I39E16I40E4E4I39E4 I40E.I39E16I40E4E16I39E.EE16E
16
3440 J=0:"!":PLAYA(J);
3450 PLAY A(20);
3460 A(0)="I40E.I39E16I40E4E4I39E4 I40E.I39E16I40E4I38O4V16=3P2E
=0I39EI40EI39E
3470 J=0:"!":PLAYA(J);
3480 NEXT
3490 PLAY""      :'Music START
```

リスト4 RYDEEN

日本音楽著作権協会(出)許諾第8972064-901号

```
 10 /*
 20 /*
 30 /*                 RYDEEN
 40 /*
 50 /*          Music by Yukihiro Takahashi
 60 /*
 70 /*                 Arranged by YMO
 80 /*
 90 /*
100 m_init()
110 for z=1 to 8
120   m_assign(z,z)
130   m_alloc(z,5000)
140 next
150 str a[256],b[256],c[256],d[256],e[256],f[256]
160 str g[256],h[256],i[256],j[256],k[256],l[256]
170 str m[256],n[256],o[256],p[256],q[256],r[256]
180 str s[256],t[256],u[256],v[256],w[256],x[256]
190 str aa[256],bb[256],cc[256],zz
200 str th,tm,tl,ft
210 str th2,tm2,tl2,ft2
220 str th4,tm4,tl4,ft4
230 str th1,tm1,bd,sd,ms
240 str lg[40],lg2[40],bm[40]
250 /*
260 dim char SYNTOM(4,10)={
270 /*   af  om  wf syc spd pmd amd pms ams pan
280      62, 15,  0,  0,  0,  0,  0,  0,  0,  3,  0,
290 /*   ar d1r d2r  rr d1l  tl  ks mul dt1 dt2 ame
300      31, 13,  4,  2,  2,  0,  0, 15,  6,  0,  0,
310      31, 16,  9,  2,  5,  3,  0,  0,  3,  0,  0,
320      31, 16,  3,  2, 10,  4,  0,  0,  3,  0,  0,
330      31, 11,  8,  3,  8,  8,  0,  0,  3,  0,  0}
340 m_vset(72,SYNTOM)
350 dim char SYNTHE(4,10)={
360 /*   af  om  wf syc spd pmd amd pms ams pan
370      58, 15,  0,  0,  0,  0,  0,  0,  0,  3,  0,
380 /*   ar d1r d2r  rr d1l  tl  ks mul dt1 dt2 ame
390      31,  5,  1, 12,  3, 41,  1,  0,  3,  0,  0,
400      31, 17,  0,  8,  2, 25,  1,  4,  3,  0,  0,
410      31, 17,  0,  0,  2, 27,  1,  3,  7,  0,  0,
420      31,  0,  0,  9,  0,  0,  0,  1,  0,  0,  0}
430 m_vset(73,SYNTHE)
440 dim char SYNBRS(4,10)={
450 /*   af  om  wf syc spd pmd amd pms ams pan
460      59, 15,  0,  0,  0,  0,  0,  3,  0,  3,  0,
470 /*   ar d1r d2r  rr d1l  tl  ks mul dt1 dt2 ame
480      18,  5,  1, 12,  3, 29,  1,  1,  1,  0,  0,
490      18, 24,  0,  8,  2, 17,  1,  1,  2,  0,  0,
500      18, 24,  0,  0,  0, 32,  0,  0,  0,  0,  0,
510      18,  0,  0,  9,  0,  0,  0,  1,  4,  0,  0}
520 m_vset(74,SYNBRS)
530 dim char CHORD(4,10)={
540 /*   af  om  wf syc spd pmd amd pms ams pan
550      59, 15,  0,  0,  0,  0,  0,  0,  0,  3,  0,
560 /*   ar d1r d2r  rr d1l  tl  ks mul dt1 dt2 ame
570      18, 12,  1,  7,  2, 32,  1,  1,  0,  0,  0,
580      18, 10,  1,  7,  3, 10,  1,  1,  6,  0,  0,
590      15, 10,  1,  7,  5, 19,  0,  1,  2,  0,  0,
600      20,  2,  1,  7,  3,  0,  0,  1,  4,  0,  0}
610 m_vset(75,CHORD)
620 dim char STRINGS(4,10)={
630 /*   af  om  wf syc spd pmd amd pms ams pan
640      60, 15,  0,  0,  0,  0,  0,  0,  0,  3,  0,
650 /*   ar d1r d2r  rr d1l  tl  ks mul dt1 dt2 ame
660      31,  7,  0,  5,  1, 30,  0,  0,  3,  0,  0,
670      20, 31,  0,  7,  0,  0,  0,  0,  3,  0,  0,
680      31,  7,  0,  5,  1, 30,  0,  1,  7,  0,  0,
690      20, 31,  0,  7,  0,  0,  0,  1,  0,  0,  0}
700 m_vset(76,STRINGS)
710 dim char SYNTHE2(4,10)={
720 /*   af  om  wf syc spd pmd amd pms ams pan
730      44, 15,  0,  0,  0,  0,  0,  0,  0,  3,  0,
740 /*   ar d1r d2r  rr d1l  tl  ks mul dt1 dt2 ame
750      31,  0,  0,  0,  0, 23,  0,  1,  7,  0,  0,
760      20, 15,  7,  5,  1,  0,  0,  4,  7,  0,  0,
770      31,  2,  0,  4,  4, 30,  0,  1,  7,  0,  0,
780      20, 15,  7,  5,  1,  0,  0,  4,  3,  0,  0}
790 m_vset(77,SYNTHE2)
800 dim char SYNBAS(4,10)={
810 /*   af  om  wf syc spd pmd amd pms ams pan
820      52, 15,  0,  0,  0,  0,  0,  0,  0,  3,  0,
830 /*   ar d1r d2r  rr d1l  tl  ks mul dt1 dt2 ame
840      28,  0,  0,  0,  0, 23,  0,  0,  3,  0,  0,
850      28,  5,  0,  8,  1,  4,  0,  0,  3,  0,  0,
860      28,  0,  0,  0,  0, 23,  0,  0,  7,  0,  0,
870      28,  5,  0,  8,  1,  4,  0,  0,  7,  0,  0}
880 m_vset(78,SYNBAS)
890 dim char BOOM(4,10)={
900 /*   af  om  wf syc spd pmd amd pms ams pan
910      62, 15,  0,  0,  0,  0,  0,  0,  0,  3,  0,
920 /*   ar d1r d2r  rr d1l  tl  ks mul dt1 dt2 ame
930      31, 15,  4,  8,  2,  0,  0, 15,  0,  0,  0,
940      31, 10,  0,  8,  2,  3,  0,  2,  0,  0,  0,
950      31, 10,  0,  8,  2,  4,  0,  0,  0,  0,  0,
960      31, 10,  0,  8,  2,  0,  0,  1,  0,  0,  0}
970 m_vset(79,BOOM)
980 dim char FLUTE(4,10)
990 m_vget(29,FLUTE)
1000 m_vset(80,FLUTE)
1010 /*
1020 /*
1030 bd="y2,23":sd="y2,14"
1040 th="{bb-aa-gg-ferrr}":th2="|:2"+th+":|":th4="|:4"+th+":|
1050 tm="{gg-fee-dd-crrr}":tm2="|:2"+tm+":|":tm4="|:4"+tm+":|
1060 tl="{dd-c>bb-aa-g<rrr}":tl2="|:2"+tl+":|":tl4="|:4"+tl+":|
1070 ft="{>aa-gg-fee-d<rrr}":ft2="|:2"+ft+":|":ft4="|:4"+ft+":|
1080 th1="{bafer}32":th1="|:4"+th1+":|
1090 tm1="{gfdcr}32":tm1="|:4"+tm1+":|
1100 lg="{g&a&b&<c>&b&a&g&f&e&d&c}16r16":lg2="|:"+lg+":|
1110 bm="c32&{d-de-efg-ga-ab-b<c}16&d8>&r32
1120 /*
1130 /*
1140 a="o7 v14 l16 y48,48
1150 b="@73q5d4e4@73q7f1r8@73frgrfreredcr
1160 c=b+">a<r@73d1&d2
1170 d=b+"dr@73a1&a2
1180 e="o6q6d4q3d8c8
1190 f="o6q7d4d8.c
1200 g="q3d8q7f4q3g8
1210 h="c8d8q6{cdc}8q3>a8<c8q7d8&d4
1220 i="a8q6{gag}8q3f8d8q6f4q3d8f8q6a2&a8{<cdc>}8q3a8g8q6a2d4f4
a2&a8g4ar<|1>g2<:||2>g1
1230 j="gg>g<rgg
1240 k="v12q8@76c4d4f4g4q7<c8c8>r4v14[*]
1250 l="@28"+e
1260 m="@28"+f
1270 n="|:"+c+d+"r2
1280 o="[$]o7v14q8@57r1r1r1r2r8"+j+"r1r1r1r1r2r8"+j+">g<rr8r4r8
"+j
1290 p=i+"[D.S.]
1300 q="[CODA]o7v12q8@73d4e4f8f8r4r2
1310 r="o7v13q2l8@73r4."+lg2+"r"+lg2+"r1v11|:4"+lg+":|r4."+lg2+
"r2v13"+lg2+"r1r2.
1320 s="v11|:6"+lg+":|rr2."+"|:3"+lg+":|rr1"+lg2+"r4"+lg+"r4v13
"+lg+"r2.v11|:"+lg+"r:|r
1330 t="v13"+lg+"r2r"+lg+"r4v11|:"+lg+"r:|"+"r2."+lg+"r|:5"+lg+
":|r
1340 u="|:4"+c+d+":|
1350 zz="r1r1r2
1360 m_trk(1,a+zz+n)
1370 m_trk(1,l+g+h+l+g+i)
1380 m_trk(1,o+k+n)
1390 m_trk(1,m+g+h+m+g+p)
1400 m_trk(1,q)
1410 m_trk(1,r)
1420 m_trk(1,s)
1430 m_trk(1,t)
1440 m_trk(1,a+u+u)
1450 /*
1460 a="o6 v13 l16 y49,48
1470 b="b-b-b-b-ffa-a-b-b-b-b-b-b-b-b-
1480 c="a-a-a-a-e-e-ffa-a-a-a-
1490 d="[$]v12@8>>"+b+b+c+"a-a-gg"+"ggggddffgga-a-aab-b-
1500 j=b+b+c+"a-a-a-a-"+c+"a-a-a-a-ggggddffggggggggggggddffggaa
bbgg<<
1510 k="v11q8@77c4d4f4g4q7<c8c8>r4v13[*]
1520 l="@76"+e
1530 m="@77"+f
1540 o="o6q7@73>b-4<d-4@73d1r8@73drerdrcrc>b-ar<
1550 q="[CODA]o6v11@76q8d4e4f8f8r4
1560 r="o4v14l8@79r4."+bm+"r1r4"+bm+"r4r"+bm+"r2rv12|:"+bm+":|r
1r2rv14"+bm
1570 s="r1r4."+bm+"r1r4"+bm+"r1r2."+bm+"r1r2"+bm+"r1r4."+bm+"r2

1580 t=o+">fr@73b-1&b-2<
1590 v=o+">b-r<@73f1&f2
1600 w="|:4"+t+v+":|
1610 m_trk(2,a+zz+n)
1620 m_trk(2,l+g+h+l+g+i)
1630 m_trk(2,d)
1640 m_trk(2,j)
1650 m_trk(2,k+n)
1660 m_trk(2,m+g+h+m+g+p)
1670 m_trk(2,q+r)
1680 m_trk(2,s)
```


▶げげーん！ やってしまった。12月号のイラスト “Merry Christmas” が “Marry” になってるぢゃないか！ 辞書まで引いて確認したのに，どこで間違ったのだらう。ああ，全国にハジをさらしてしまった。描いた当時の不安定な精神状態の表れか？ ま，黙ってりゃわかんねーか（これでも英検3級）。 味野 真一(23)岡山県

```
 1690 m_trk(2,a+u)
 1700 m_trk(2,w)
 1710 /*
 1720 a="o4 @v120 q6 l16 y50,48
 1730 b="@78>b-4a4<
 1740 c="ddddd<d>af":c=c+c
 1750 d=">b-b-b-b-b-<b-fd":e=d+d
 1760 f=">ggggg<gd>b-<
 1770 g="@8q2dddddddddddddddd
 1780 h="cccccccccccccccc
 1790 i="c<c>c<ccccc>
 1800 j=zz+"[$]|:"+a+b
 1810 k="|1"+i+":||2"+i+"r2
 1820 l="o4 v11 l8 q6 @74
 1830 m="b-b-fa-b-r4b-
 1840 n="a-a-e-fa-r4
 1850 o=l+m+"&"+m+n+"g8&ggdfga-ab-&
 1860 p=m+"&"+m+n+"a-&"+n+"a-ggdfgr4gggdfgabg
 1870 q="@76o6v12q8g4b-4<c4d4ggr4@v120[D.S.]
 1880 r="v9>b-4a4<ddr4r2r1r1r1
 1890 s="d<d>d<d>d<d>dd<d>d<d>dd<d>d<d>
 1900 t="f<f>f<f>f<f>ff<f>f<f>ff<f>f<f>
 1910 u="@8o5v12q4l16|:2"+s+s+t+t+":|"+s+s+t+"f<f>f<f>f<f>ff
 1920 v=a+b+c+c+e+f+b+c+c+e+d
 1930 w="@80d4{efede}4@80fd>a<dfrerfd>a<dfrer@80fr{fef}8grfrered
cr
 1940 x="d>b-fb-<drcr
 1950 aa="o7 v14 q7 l16|:7"+w+">a<r@80"+x+x+x
 1960 bb=w+"dr@80ar{ab-a}8gr{gag}8fr{fgf}8defgar{ab-a}8{gagfgf}4
:|
 1970 m_trk(3,j+c+c+e+f)
 1980 m_trk(3,b+c+c+e+d+b)
 1990 m_trk(3,g+g+g+g+g+g)
 2000 m_trk(3,h+k)
 2010 m_trk(3,o+p+q+r)
 2020 m_trk(3,u)
 2030 m_trk(3,v)
 2040 m_trk(3,aa+bb)
 2050 /*
 2060 a="o4 v11 q8 l4 y51,48
 2070 b="e@75f.g.a.@75f.ec
 2080 c="[$]|:v11@75d"+b+"d1&d2f"+b+"d2f2<d2fe>
 2090 d="rb-2&b-8<c8&c1>rb-2&b-8<c8&c2>fe8f16e16
 2100 e="d.f.g.a.d2r.dfa8&a2:|fg
 2110 f="e-2.&e-8d8&d1
 2120 g="@74"+f+"f2.&f8f8&f2f.e-8&"+f+"f2.&f8f8&f1f2.&f8f8&f1f1<
c8c8>r[D.S.]r2
 2130 h="|:15r1:|
 2140 i="r2|:4@75d"+b+"d1&d2@75d"+b+"d2f2a2:|
 2150 j="@74d4e4@74fd>a<gd>a<ad>a<@74fd>a<e>a<c>a<
 2160 k="o5l8|:4"+j+"d2rcr>b-4<fd>b-<"+j+"d>b-fb-<fd>a<dafdf:|
 2170 m_trk(4,a+zz+c+d+e)
 2180 m_trk(4,g)
 2190 m_trk(4,h)
 2200 m_trk(4,a+i)
 2210 m_trk(4,k)
 2220 /*
 2230 a="o4 v11 q8 l4 y52,32
 2240 b="@75p1>b-a<@75p1d.e.f.@75p1d.c>a<
 2250 c="[$]|:v11"+b+">b-8<c>b-ag8g2<"+b+">b-2<d2f2b-a
 2260 d="rd2&d8d8&d1rd2&d8d8&d2fe8f16e16
 2270 e="d.>b-.<c.d.>b2<r8>b-.&b-1<:|>b-<c
 2280 f="f2.&f8f8&f1
 2290 g="@74p1"+f+"a-2.&a-8g8&g2g.f8&"+f+"a-2.&a-8a-8&a-1g2.&g8g
8&g1g1<g8g8>r[D.S.]r2
 2300 h="o5 @v123 l16 @72
 2310 i="v0y52,0p1rr@v123"+tl2+"r"+tl+"r4"+tm2+"r"+tm2+"rrr"+th4
+th+"rr"+tm4+"rr
 2320 j=tl2+"r"+tl2+"rrr"+ft4+"rrr"+tl2+"r"+tl2+"r"+tm1+tl2
 2330 k=tm2+tl2+ft2+ft+"rrr"+tl+"rr"+tl+"r"+tl2+"rr"+tm+"r"+tm+"
rr
 2340 l=tl+"r"+tm+th+tm+th+"rr"+tm+"r"+tm2+"r"+tm2+"rrr
 2350 m=tl2+th+tl2+th+tl2+th+tl2+th+tl+tm2+"r
 2360 n=ft+tl+ft+"r"+ft+"rr"+ft+"r"+ft2+"r"+ft+"r
 2370 o="rr"+th4+"r"+tm2+tm1+tm1+tm+"r"+tl
 2380 p=tl4+tl+ft2+tl2+tl+"r"+tl+tm+"r"+tl+"r
 2390 q="r"+tm+tl2+ft+"r"+tm+"rrr"+tm+"r"+tm2+"r"+tm
 2400 r="r2|:8"+b+"@75p1>b-8<c>b-ag8g2<"+b+"@75p1>b-2<d2f2:|
 2410 m_trk(5,a+zz+c+d+e)
 2420 m_trk(5,g+h)
 2430 m_trk(5,i)
 2440 m_trk(5,j)
 2450 m_trk(5,k)
 2460 m_trk(5,l)
 2470 m_trk(5,m)
 2480 m_trk(5,n)
 2490 m_trk(5,o)
 2500 m_trk(5,p)
 2510 m_trk(5,q)
 2520 m_trk(5,p)
 2530 m_trk(5,o)
 2540 m_trk(5,p)
 2550 m_trk(5,q)
 2560 m_trk(5,a+r)
 2570 /*
 2580 a="o4 v11 q8 l4 y53,64
 2590 b="@75p2>b-<d-@75p2d.e.f.@75p2d.c>a<
 2600 c="[$]|:v11"+b+">b-8<c>b-<c>b-8b-2<"+b+">b-2<d2b-2<dc>
 2610 d="rf2&f8g8&g1rf2&f8g8&g2fe8f16e16
 2620 e="d.>b-.<c.d.>b2<r8c.&c1:|de
 2630 f="b-2.&b-8b-8&b-1
 2640 g="@74p2"+f+"<c2.&c8c8&c2>b.b-8&"+f+"<c2.&c8c8&c1c2.&c8>b8
&b1<c1[*]g8g8r[D.S.]
 2650 h="[CODA]"+h
 2660 i="y53,0p2"+tm+"r|:20"+tm+":|r"+tm4+"r4"+tl2+"r4"+ft+"rr"+
ft+ft2+tl+tl2
 2670 j=tm+tm2+"r"+th2+th4+th1+th4+"r"+tm2+tm4+"r"+tl4+"r"+ft4+"
rrr
 2680 k=tl+"rr"+tl+tl2+"r"+tm+"r"+tm+"r"+tm2+"rr"+tm+"rr"+tm+"rr
 2690 l=tm2+"r"+tm2+"r"+th2+"rr"+th+"r"+th2+"rr"+th+"r"+th2+"rr"
+tm+"rrrr
 2700 m=tl+tm+tl+tm2+th4+th+tm2+tl2
 2710 n=tl+"rr"+tl+tl2+"r"+ft+"r"+"|:13"+ft+":|"+"rr"+ft2+tl+tm2
+"rr
 2720 o="|:"+ft+tl+tm+tm+"r:|"+ft+tl+tm2+tm+"r"+tm
 2730 p="|:5r"+tm+":|"+tm+tl+tm+tl2+"r
 2740 q="p3"+tm2+tm+"r"+tm2+tm+"r
 2750 r="|:8"+b+"@75p2>b-8<c>b-<c>b-8b-2<"+b+"@75p2>b-2<d2f2:|
 2760 m_trk(6,a+zz+c+d+e)
 2770 m_trk(6,g+h)
 2780 m_trk(6,i)
 2790 m_trk(6,j)
 2800 m_trk(6,k)
 2810 m_trk(6,l)
 2820 m_trk(6,m)
 2830 m_trk(6,n)
 2840 m_trk(6,o)
 2850 m_trk(6,p)
 2860 m_trk(6,n)
 2870 m_trk(6,o)
 2880 m_trk(6,q)
 2890 m_trk(6,a+r)
 2900 /*
 2910 a="o3 @v126 q8 y54,48
 2920 b="@78l4>b-a<@78d.e.f.@78d.cf
 2930 c="[$]|:"+b+">b-2r8a8r8g8&g2<"+b+">b-1b-2b-a<
 2940 d="l8>gfg4&gb-<cd&d2
 2950 e=d+"dc>a4<"+d+"d4c4>b-4.g4.a4.b-4.b2c2&c<dc>e&e2<:|c4c4
 2960 f=">b-4b-4b-b-rb-&b-b-b-4b-b-rb-<
 2970 g=f+">a-4a-4a-a-rg&ggg4ga-ab-&<"+f+">a-4a-4a-a-ra-&a-a-a-4
a-a-ra-<
 2980 h=">g4g4ggrg&ggg4gabg<c4c4ccc4ccr4[D.S.]
 2990 i=">b-b-r2ra<
 3000 j=">b-4a4<|:4>ddr2r>a<dd<r2.b-b-r2rf|1"+i+":|"+"|2"+i+":|"
+"|3"+i+":|"+"|4>b-b-<r4
 3010 k="|:8"+b+"@78>b-2r8a8r8g8&g2<"+b+"@78>b-1b-2<:|
 3020 m_trk(7,a+zz+c+e)
 3030 m_trk(7,g+h)
 3040 m_trk(7,j)
 3050 m_trk(7,k)
 3060 /*
 3070 a="v15 q7 l16 @59 y56,0 y3,3
 3080 b="o1p2g"
 3090 c="o7p1g"
 3100 d=b+"r"+c+b
 3110 e=bd+d+sd+d
 3120 f=bd+b+"r"+bd+c+b+sd+d:f=e+f
 3130 g=sd+d+sd+d
 3140 h=bd+b+"r"+sd+c+sd+b+sd+b+sd+"r"+sd+c+b
 3150 i=bd+b+"r"+sd+c+sd+b
 3160 j="|:"+sd+b+sd+"r"+sd+c+b+":|
 3170 k=sd+b+sd+"r"+sd+c+sd+b+sd+b+sd+"r"+sd+c+b
 3180 l=a+"|:10"+d+":|"+g+"|:15"+e+":|"+h
 3190 m="|:7"+f+":|"+e+g+"|:15"+e+":|
 3200 n=h+f+e+"|:"+i+":|
 3210 o="|:5"+f+":|"+e+i+sd+b+sd+"r"+sd+c+sd+b
 3220 p="|:11"+f+":|"+e+g+"|:14"+e+":|"
 3230 q="|:"+i+":|"+bd+b+sd+"r"+sd+c+sd+b+sd+b+sd+"r"+sd+c+b
 3240 r="|:3"+f+":|"+e+h
 3250 s="|:3"+f+":|"+e+g
 3260 t="|:15"+e+":||:"+sd+b+"r"+sd+c+b+":||:7"+f+":|
 3270 u=h+k+"|:11"+f+":|"+bd+b+"r"+bd+c+b
 3280 v=d+g+"|:60"+bd+d+":|"+"|:"+j+k
 3290 w="|:7"+e+":|"+g+"|:6"+e+":|"+h
 3300 x=k+"|:7"+e+":|"+bd+b+"r{"+sd+c+sd+b+sd+"r}8"+sd+b+sd+"r"+
sd+c+b+"|:7"+e+":|
 3310 aa=i+i+"|:7"+e+":|"+j+"|:6"+e+":|"+bd+b+"r"+sd+c+b+"|:3"+s
d+b+"r"+sd+c+b+":|
 3320 bb="|:7"+e+":|"+h+"|:6"+e+":|"+i+i
 3330 cc=bd+b+sd+"r"+sd+c+sd+b+sd+b+sd+"r"+sd+c+b
 3340 m_trk(8,l)
 3350 m_trk(8,m)
 3360 m_trk(8,n)
 3370 m_trk(8,o)
 3380 m_trk(8,p)
 3390 m_trk(8,q)
 3400 m_trk(8,r)
 3410 m_trk(8,s)
 3420 m_trk(8,t)
 3430 m_trk(8,u)
 3440 m_trk(8,v)
 3450 m_trk(8,w)
 3460 m_trk(8,x)
 3470 m_trk(8,aa)
 3480 m_trk(8,bb)
 3490 m_trk(8,cc)
 3500 /*
 3510 /*
 3520 m_tempo(143)
 3530 m_play()
 3540 /*
 3550 /*
 3560 for z=0 to 1065000:next
 3570 for y=0 to 120
 3580   SYNTHE(4,5)=SYNTHE(4,5)+1:m_vset(73,SYNTHE)
 3590   SYNBRS(4,5)=SYNBRS(4,5)+1:m_vset(74,SYNBRS)
 3600    CHORD(4,5)= CHORD(4,5)+1:m_vset(75,CHORD)
 3610   SYNBAS(2,5)=SYNBAS(2,5)+1
 3620   SYNBAS(4,5)=SYNBAS(4,5)+1:m_vset(78,SYNBAS)
 3630    FLUTE(4,5)= FLUTE(4,5)+1:m_vset(80,FLUTE)
 3640 z1=2000-y*15
 3650 for z=0 to z1:next
 3660 next
 3670 m_stop()
```

▶３歳の息子が上海の１面をクリアしてしまった。将来が楽しみ，不安，ワクワク，ドキドキ……。
吉田　早苗(27)埼玉県

★(で)のショートプロぱーてぃ　その5

お年玉(?)はユーティリティ

Komura Satoshi　古村　聡

ショートプログラムといっても，いつもBASICばかりとか，ゲームばかりだというわけじゃありません。今月はちょっとしたユーティリティを2本紹介します。X68000での外部関数作成の参考にもなるでしょう。

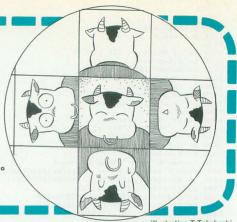



illustration:T.Takahashi

この原稿がOh!Xに載る号は新年1月号なわけですね。

A HAPPY NEW YEAR!

ああ，めでたいっ！

んでもって発売日はクリスマスにもむちゃくちゃ近いわけなんですよね。

MERRY X'mas!

すごくめでたいっ！

というわけで，この連載向きのいい季節になってきて，ぱーっと盛大に特別企画でもいこうかー……と思っていたのですが，いま，私がこれを書いている今日はなんと文化の日の11月3日。実をいうとこれが1月号だなんてことはすっかり忘れてしまっていたのですねー。

というわけで，お年玉やクリスマスプレゼントというよりは，ほのかに文化の香りといった感じの「ユーティリティ2本組」になってしまいました。ま，こんなのもたまにはいいですよね（待てよ……，ってことはもしかして2月発売のOh!Xにはバレンタインデーの話をしなきゃならんのか……やだなあ)。

まずはメニューから

さっそく，プログラムのほうにいってしまいましょう。まず今月のユーティリティ1本目。

●MENU.bas For CZ-FB0X

三重県　水谷　潔

X68000のVSやPC98のMS-DOS（最近のやつね）とかを使ったことのある人ならわかると思いますが，ディスクに何本もプログラムが詰まっている場合，DOSやBASICが立ち上がったときにプログラムのメニューが出てきて実行するプログラムを選べるようになっていてくれるとすごく便利なんですよね。

このプログラムはそれをX1で，しかもオールBASICのプログラムでやってしまい，そのうえ新旧turbo 3つのBASICに対応しているという，小さいけれど縁の下の力持ち，というプログラムなのです。

まずこのプログラムを打ち込んで，

SAVE“MENU.bas”

として，次にBASICの“Startup.bas”(BASICのいちばん最初に実行するプログラム）の最後に「Run“MENU.SYS”」と付け加えます。

で，電源を入れ直すなりリセットするなりしてやればあーら不思議（おいおい)，BASICとマシン語のプログラム（ASCIIファイルは除く）のメニューがおぞぞぞっ，と出てきてカーソルキーで選んでリターンキーをちょいと押せば，そのプログラムが走り出してくれるのです。

そうそう，ただ，ちょっとプログラムが

MENU.basの実行例

小さいんで階層ディレクトリはサポートしていません。ま，BASICでわざわざ階層使う人もいないと思うから，これでいいんですよね。あと，表示できるファイル数は最大で75個です。

プログラム的にいえばこのユーティリティ，DSKI$命令（セクタ単位でディスクを読む命令）でディスクのディレクトリ領域を読んできてRUNさせるだけという単純なプログラムです。オールBASICで書かれているという比較的初心者向けのプログラムなのですが，これだけ便利で実用的なプログラムになっているのにはBASICにもそして作者の水谷さんにも頭が下がります。

X1ユーザーでBASICしかわからない人でも，もちろんそれ以外の人でも，ぜひプログラムリストのほうを読んでみてください。セクタやディレクトリに関しても囲みで説明して，BASICしかわからない人でもばっちりわかるようにしときますからね。

まわるまわる……

次のプログラムはX68000用のBASIC外部関数です。

●GMOVE.FNC For X-BASIC

中川　勝豊

これ，もともとコマンドライン用のプログラムとして投稿されてきたのですが，こちらでBASIC用の関数に直したものです。

ディスク管理

ディスクにはいろいろデータが書き込まれているわけですけど，ディスクを使うときにフォーマットしますよね。フォーマットするときにディスクドライブがディスケットにプログラムを入れる枠のようなものを設定します。その枠のようなものをレコードといって640レコードをひとつのディスケットに作っていき，ひとつ目のレコードからレコード1，2，3と番号がついています。

んで，レコード16からがX1 BASICのファイル名が書いてある部分です。ひとつのファイルにつき32バイトが割り当てられていて，その32バイトのうち，2バイト目から14バイト目までがファイル名，15バイト目から17バイト目までが拡張子（たとえばMENU.BASならMENUがファイル名，BASが拡張子）に割り当てられていて文字コードがそのまま入っているわけですね。

んじゃ，レコードから内容を読んで，そのまま文字列変数に入れてRUNしてやればそのファイルを実行することができるというわけです。

このプログラムはグラフィックに描かれた絵を90度回転させたり左右反転させたりする外部関数です。まあ，口でごちゃごちゃいうより実行してもらったほうが絶対インパクトがありますからぜひそうしてください（画面消去の「しわわ〜」っと消えるのが感動です）。

X-BASICの外部拡張関数っていうのはマシン語がわかっている（つまり，プログラムのリストが手許にあれば）なら簡単に自分で拡張できるようになっていますから，このプログラムを参考にしてぜひとも自分で拡張関数を作ってこのコーナーに投稿してください。

で，この関数のBASICへの組み込み方なのですけど，まず，

ED　GM.S

としてエディタを立ち上げます。リストをそのまま打ち込んでいって，終わったらESCキーを押してからEを押してエディタを抜けます。

で，

AS　GM

を実行してさらに，

LK　GM

と実行してください。ASを実行したときに，

"No Fatal error(s)"

と出てこなかったら打ち込み間違いがあります。"ED GM.S"へ戻って修正してください。

さて，ここまでこれたら完成したプログラムをBASICにくっつけます。まず，

COPY GM.X ¥BASIC2¥GM.FNC

と打ってBASICのところまで持っていきます。そして，最後のツメです。

ED　¥BASIC2¥BASIC.CNF

としてエディタに入ってください。

```
FREE  =256
WIDTH =64
BEEP  =ON
CAPS  =OFF
FUNC  =AUDIO
FUNC  =GRAPH
   ⋮
FUNC  =IMAGE
```

こんな風に書いてあるはずです。最後の行に，

FUNC　=GM

と付け加えれば，やっと完成です。サンプルプログラムで遊ぶのもよし，自分で描いた絵をぐるぐる回して遊ぶのもよし。感動しますよぉ。

おっと，忘れてた。この関数はBASICのリストに，

GMOVE（引数）

とやって使います。引数の値は整数（つまりINTってやつですね）で，

0　グラフィック画面消去
1　グラフィック画面ON
2　グラフィック画面左右反転
3　無効
4　テキストのパレット0の色を青に
5　グラフィック画面を180度回転
6　テキストのパレット0の色を戻す
7　グラフィック画面を左90度回転
8　グラフィック画面上下反転
9　グラフィック画面を右90度回転

となっています。SCREENモードは1，X，1，1。X＝1〜3です。

今月の感想

GMのほうは私がちょっとばっかり手を入れてしまいましたが，今月はどっちもよくできたユーティリティなので私は感動してしまいました。このぐらいよくできたショートがくればこれからもいつでも載せていきたいと思っています。「ゲームだけがショートじゃない」ってハガキを送ってきてくれた人もいたことですしね。

ただ，ユーティリティだと，やっぱり説明的な文章が多くなっちゃいましたねー。いかがでしたか？

プログラムの説明がこれからもあったほうがいいとか，ちょっと読みにくいよー，とかいろいろ感想待ってます（ほとんど「連載・毎月が微調整」って感じだなー）。

んじゃ，そーゆーことでまた来月。

よくわかりませんが，回転してます

条件分岐の話

今回のX68000用のプログラムで私が手を入れたのはパラメータの判定の部分でした。パラメータがデータレジスタD5に入っていてその内容をキーにしていろいろな処理内容に飛ぶ，という処理をするのですが編集室に送られてきたプログラムはこんなふうになっていました。

```
    cmp.b   #7,d5
    beq     l_roll     *7なら90度回転
    cmp.b   #9,d5
    beq     r_roll     *9なら−90度回転
    cmp.b   #4,d5
    beq     b_set      *4なら背景色SET
    cmp.b   #6,d5
      ⋮
    cmp.b   #0,d5
    beq     g_clr.     *0なら画面クリア
    rts                *メインへ帰る
```

確かにこういうコンペアと条件分岐がずらーっと並んでいるだけでもちゃんと動きますし，読んでもわかりますからべつに悪いとはいいませんが，BASICにもIF〜THEN GOTOの羅列よりもON〜GOTO命令が使われるようにマシン語でも定石としてこんなふうに書くのがふつうです。

```
    lea     JMPTBL,a5
    lsl.l   #2,d5
    jmp     0(a5,d5.l)
JMPTBL:
    bra.w   g_clr      *0なら画面クリア
    bra.w   gvon       *1ならGVON
    bra.w   g_side     *2なら左右反転
      ⋮
    bra.w   r_roll     *9なら−90度回転
```

68000の豊富なアドレッシングモードはこういうときのためにあるのです。この方法は昔はゲームのキー入力なんかでよく使っていました。なぜなら前の方法ではD5の内容によって処理速度が微妙に違ってしまいますが後者ではD5の内容によらず処理速度は一定だからです。

また，こうすることによってプログラムのサイズをわずかながら，縮める効果もあります。このプログラムでは条件分岐の範囲がワードですのでひとつ当たり4バイト必要ですがうまくショートブランチの範囲内にすべてのルーチンを収めるようにしてやればジャンプテーブルの大きさが半分（もちろん，lsl.l #1,D5にする）になるのでさらに小さくなります。またこのプログラム自体もまだインデックスをうまく使ったりいろいろひねればまだまだ小さくなることでしょう。

リスト1

```
10 '  SAVE"ﾒﾆｭｰﾀﾞｼﾖﾝ" '          1989/3       by TADOMAME
20 '
30  WIDTH 80,25:INIT:CLS 4:KLIST 0:CONSOLE 0,25' <--- Turbo
40 'WIDTH 80:INIT:CLS 4 '                       <--- X1
50 DEFINT A-Z
60 DIM BB$(2),NM$(75),BB(75)
70 MAX=1:REC=16:X=0:Y=0
80 BB$(1)="Bin:":BB$(2)="Bas:"
90 '
100 DEVI$"0:",REC,FA$,FB$
110 FOR I=0 TO 3:DT$=MID$(FA$,I*32+1,32):GOSUB 420:NEXT
120 FOR I=0 TO 3:DT$=MID$(FB$,I*32+1,32):GOSUB 420:NEXT
130 REC=REC+1:GOTO 100
140 '
150 X=0:Y=0:NN=1:PLAY"O4C0"
160 LOCATE X*26,Y:CREV 1:PRINT BB$(BB(NN));NM$(NN);:CREV 0
170 KEY0,"":K$=""
180 K$=INKEY$:IF K$="" THEN 180
190 IF K$=" " OR K$=CHR$(13) THEN 260
200 IF K$="2" OR K$=CHR$(31) THEN 310
210 IF K$="4" OR K$=CHR$(29) THEN 330
220 IF K$="6" OR K$=CHR$(28) THEN 350
230 IF K$="8" OR K$=CHR$(30) THEN 370
240 PLAY"O4C0":GOTO 160
250 '
260 CLS:LOCATE 10,10:PRINT"ｼﾊﾞﾗｸ ｵﾏﾁｸﾀﾞｻｲ";CHR$(11)':END <--- ﾃﾞﾊﾞｯｸ ｼﾞ  [']  ｦ  ﾄﾙ
270 'KEY0,CHR$(30,29):CLS:PRINT "RUN";CHR$(34);NM$(NN);CHR$(34);:NEWON
280 IF BB(NN)=1 THEN KEY0,"LO.M"+CHR$(34)+NM$(NN)+CHR$(34)+",R"+CHR$(13):END
290 KEY0,"R."+CHR$(34)+NM$(NN)+CHR$(34,13):END
300 '
310 IF Y<24 AND NN+3<MAX THEN YY=Y+1:GOTO 390 ELSE 240
320 '
330 IF X>0 THEN XX=X-1:GOTO 390 ELSE 240
340 '
350 IF X<2 AND NN+1<MAX THEN XX=X+1:GOTO 390 ELSE 240
360 '
370 IF Y>0 THEN YY=Y-1:GOTO 390 ELSE 240
380 '
390 LOCATE X*26,Y:PRINT BB$(BB(NN));NM$(NN);:X=XX:Y=YY:NN=Y*3+X+1
400 PAUSE 1:GOTO 160
410 '
420 BB=ASC(LEFT$(DT$,1))AND &H83:IF BB=1 OR BB=2 ELSE IF BB=&H83 THEN RETURN 150
ﾅﾗ ｵﾜﾘ
430 IF MID$(DT$,2,8)="Start up" OR MID$(DT$,15,3)="Sys" THEN RETURN
440 BB(MAX)=BB:NM$(MAX)=MID$(DT$,2,13)+"."+MID$(DT$,15,3)
450 LOCATE X,Y:PRINT BB$(BB);NM$(MAX)
460 MAX=MAX+1:IF MAX=76 THEN RETURN 150
470 X=X+26:IF X=78 THEN X=0:Y=Y+1
480 RETURN
490 ' --- ｺﾚﾃﾞ ｵﾜﾘ ---
```

リスト2

```
=================== gm3.s ===================
  1: *
  2: *       GRAPHIC EFECT VERSION 4.00
  3: *
  4: *       PRGRAMED  By Show   H1.10.20
  5: *       BASICver. By DEC    H1.11.03
  6:
  7:    .GLOBL  _gmove       *←これはコンパイル用ね。
  8:
  9:
 10:         *
 11:         * Information Table
 12:         *
 13:         .text
 14:
 15:         .dc.l   F_init      *←BASICの諸々の
 16:         .dc.l   F_run       *  テーブルたち
 17:         .dc.l   F_end       *  このへんは全部共通だから
 18:         .dc.l   F_exit      *  一つ憶え、と。
 19:         .dc.l   F_break
 20:         .dc.l   F_ctrlD
 21:         .dc.l   F_dmy1
 22:         .dc.l   F_dmy2
 23:         .dc.l   F_token
 24:         .dc.l   F_parTbl
 25:         .dc.l   F_exec
 26:         .dc.l   0,0,0,0,0
 27:
 28:         * dummy *←でも初期設定はない。
 29: F_init:
 30: F_run:
 31: F_end:
 32: F_exit:
 33: F_break
 34: F_ctrlD:
 35: F_dmy1:
 36: F_dmy2:     rts
 37:
 38:             * TOKEN table
 39: F_token:    dc.b  'gmove',0 *←コマンドの名前（小文字で！ ）
 40:             dc.b  0
 41:
 42:       .even
 43:
 44:       *
 45:       * param table
 46:       *
 47:
 48: F_parTbl:  dc.l    gmove_par
 49:
 50:
 51:       *
 52:       * parameter ID table
 53:       *
 54:
 55: gmove_par: dc.w  $02        *引き数は整数（省略不可）
 56:            dc.w  $ffff      *戻り値なし
 57:
 58:       *
 59:       *EXEC TAble
 60:       *
 61:
 62: F_exec:     dc.l     gmove_func
 63:
 64: iocs        macro    pr1      *IOCSコールのマクロ定義
 65:             move.l   #pr1,d0 *（IOCS.MACをつかわないから）
 66:             trap     #15
 67:             endm
 68:       .even
 69:
 70: gmove_func move.l 12(sp),d5 *GMOVE（BASIC）のエントリ
 71:             bsr    gminit
```

```
 72:            clr.l  d0
 73:            rts
 74:
 75: _gmove    move.l   8(a6),d5 *ここはBtoCに作らせてしまいました
 76:           BRA      L4       *手抜き手抜き。
 77: L5:
 78:           MOVE.L   8(A6),D5
 79:           bsr      gminit
 80:           MOVE.L   D0,8(A6)
 81: L6:
 82:           UNLK     A6
 83:           RTS
 84: L4:
 85:           LINK     A6,#0
 86:           BRA      L5
 87:
 88: *
 89: *    プログラム選択
 90: *
 91:
 92:
 93: gminit: subq.l   #1,a2
 94:         sub.l    a1,a1     *SUPERへ
 95:         iocs     $81
 96:         move.l   d0,a6     *A6はSUPERのスタック
 97:
 98: start   move.l #$C00000,a0    ******
 99:         move.l #$C00000+1022,a1   *
100:         move.l #$C7FFFE,d0       *+-90度回転共通部分
101:         move.l #$C80000-1024,d1   *
102:         move.l #511,d7         *******
103:         move.l #$C00000,a3 * 上下、左右180度共通DATA
104:
105:         cmp.l  #10,d5
106:         bcc    pend
107:         lea    JMPTBL,a5    *ここが例の縮めた部分。
108:         lsl.l  #2,d5        *BASICのON gotoみたいなもんです。
109:         jmp    0(a5,d5.l)
110:
111:
112: JMPTBL: bra.w  g_clr        * 0なら画面クリアー
113:         bra.w  gvon         * 1ならGVON
114:         bra.w  g_side       * 2なら左右反転
115:         bra.w  pend         * 3は無効
116:         bra.w  b_set        * 4なら背景色SET
117:         bra.w  h_roll       * 5なら180度回転
118:         bra.w  b_reset      * 6なら背景色RESET
119:         bra.w  l_roll       * 7なら90度回転
120:         bra.w  g_ud         * 8なら上下反転
121:         bra.w  r_roll       * 9なら-90度回転
122:
123: pend:   move.l a6,a1        * ユーザーモードへ
124:         iocs   $81
125:         rts                 * 作業終了。BASICへ
126: *
127: *    背景色SET
128: *
129: b_set move.l #%0000000_0000000_00000_00000_01110_0,D2
130: *            0000000 0000000 GGGGG RRRRR BBBBB 0  背景色
131:         move.l #0,d1
132:         bra    b_sub
133:
134: *
135: *    背景色RESET
136: *
137:
138: b_reset move.l #$0,d2
139: b_sub   iocs   $13
140:         bra    pend
141:
142: *
143: *    GVON
144: *
145: gvon    move.w #$10C,d1
146:         iocs   $10
147:         move.l #$E82600,a1
148:         move.w #%00101111,(a1)
149:         bra    pend
150:
151: *
152: *    上下反転ルーチン
153: *
154: *move.l #$C00000,A3*上下,左右180度共通DATA 定義済
155:
156: g_ud    move.l #$C7FC00,a4
157: udlp1   move.l #512,d5
158: udlp2   move.w (a3),d3
159:         move.w (a4),(a3)+
160:         move.w d3,(a4)+
161:         cmp.l  #$C40000,a3
162:         bge    start
163:         subq.l #1,d5
164:         bgt    udlp2
165:         sub.l  #2048,a4
166:         bra    udlp1
167:
168: *
169: *    左右反転ルーチン
170: *
171: *move.l #$C00000,A3 *上下,左右180度共通DATA定義済
172:
173: g_side  move.l #$C003FE,a4
174: gslp    move.w (a3),d3
175:         move.w (a4),(a3)
176:         move.w d3,(a4)
177:         add.l  #1024,a3
178:         add.l  #1024,a4
179:         cmp.l  #$C7FFFE,a3
180:         blt    gslp
181:         sub.l  #$7FFFE,a3
182:         sub.l  #$80002,a4
183:         cmp.l  a3,a4
184:         bgt    gslp
185:         bra    pend
186:
187:
188: *
189: *    180度回転ルーチン
190: *
191: *move.l #$C00000,a3 *上下,左右180度共通DATA 定義済
192:
193: h_roll  move.l #$C80000,a4
194: hrlp    move.w (a3),d3
195:         move.w -(a4),(a3)+
196:         move.w d3,(a4)
197:         cmp.l  a3,a4
198:         bgt    hrlp
199:         bra    pend
200:
201: *
202: *    画面クリアールーチン
203: *
204:
205: g_clr  moveq.l #0,d0
206:        moveq.l #15,d1
207: gclp    move.w d0,(a3)
208:         add.l  #30,a3
209:         cmp.l  #$C80000,a3
210:         blt    gclp
211:         sub.l  #$80000,a3
212:         subq.l #1,d1
213:         bgt    gclp
214:         bra    pend
215:
216: *
217: *    90度回転ルーチン
218: *
219: *   move.l  #$C00000,a0   ******
220: *   move.l  #$C00000+1022,a1   *
221: *   move.l  #$C7FFFE,d0        *この部分共通 定義済
222: *   move.l  #$C80000-1024,d1   *
223: *   move.l  #511,d7        *******
224:
225: l_roll  move.l a2,d5         *   A2待避
226: lrlp1   ove.l  d7,d6
227:         move.l a0,a2
228:         move.l a1,a3
229:         move.l d0,a4
230:         move.l d1,a5
231: lrlp2   move.w (a2),d2
232:         move.w (a3),d3
233:         move.w (a4),d4
234:         move.w (a5),(a4)
235:         move.w d3,(a2)
236:         move.w d4,(a3)
237:         move.w d2,(a5)
238:         addq.l #2,a2
239:         subq.l #2,a4
240:         add.l  #$400,a3
241:         sub.l  #$400,a5
242:         subq.l #1,d6
243:         bne    lrlp2
244:         subq.l #2,d7
245:         bcs    lrend
246:         add.l  #1024+2,a0
247:         add.l  #1024-2,a1
248:         sub.l  #1024+2,d0
249:         sub.l  #1024-2,d1
250:         bra    lrlp1
251: lrend   move.l d5,a2
252:         bra    pend
253:
254: *
255: *    -90度回転ルーチン
256: *
257: *   move.l  #$C00000,a0   ******
258: *   move.l  #$C00000+1022,a1   *
259: *   move.l  #$C7FFFE,D0        * この部分共通 定義済
260: *   move.l  #$C80000-1024,d1   *
261: *   move.l  #511,d7        *******
262:
263: r_roll  move.l a2,d5         *   A2待避
264: rrlp1   move.l d7,d6
265:         move.l a0,a2
266:         move.l a1,a3
267:         move.l d0,a4
268:         move.l d1,a5
269: rrlp2   move.w (a2),d2
270:         move.w (a3),d3
271:         move.w (a4),d4
272:         move.w (a5),(a2)
273:         move.w d2,(a3)
274:         move.w d3,(a4)
275:         move.w d4,(a5)
276:         addq.l #2,a2
277:         subq.l #2,a4
278:         add.l  #$400,a3
279:         sub.l  #$400,a5
280:         subq.l #1,d6
281:         bne    rrlp2
282:         subq.l #2,d7
283:         bcs    rrend
284:         add.l  #1024+2,a0
285:         add.l  #1024-2,a1
286:         sub.l  #1024+2,d0
287:         sub.l  #1024-2,d1
288:         bra    rrlp1
289: rrend   move.l d5,a2
290:         bra    pend
291:      .end
```

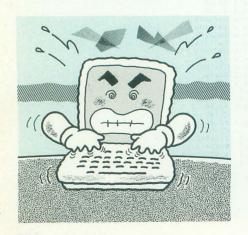
▶おっ！ 本の「べりっ」という音が消えているではないか！ 見た目はそんなに変わってないのに，なんとなく得した気分だぁ。 栗坂 明(20)埼玉県

1000bitプロセッサの世界

RISC, RISCの大合唱

パソコンの世界ではまだその波は押し寄せてきていないようですが，パソコンより1クラス上のマシンであるワークステーションの世界では，いまやプロセッサにはRISC（簡単な命令だけからなる命令セットを用意して滅茶苦茶速いスピードを実現するアーキテクチャとでもいいましょうか）を採用するのが，打ち消すことのできない流れのようです。確かに10MIPS程度でふーふーいっているCISCに対して，RISCが「20MIPS，30MIPSは当たりまえー！」とカナキリ声を上げているのですから，ついそっちに手を出してしまうのが当然な話といえましょう。マクロに見た場合，たとえばソフトウェアのライフサイクルという面から見ても，本当にそっちが速いかどうかは，かなり問題だと思われますが。

こう，RISC，RISCと大合唱されると，最初，そう8年ぐらい前に論文を読んだときには，これはすごいと思った気持ちもさすがにどっかにいってしまい，いったいなんなんだ，このブームは，という気持ちになってしまいます。まあとにかく速いことはいいことなのでしょう。僕自身もネットワークに恵まれている（妙な言葉だ）ので，仕事に取りかかり始めるときは自分の横1メートルに置いてある68系のプロセッサを使っていても，ちょっと時間のかかるプログラムを走らせる場合には，実物をろくに見たこともない遠くにあるワークステーションの中に収まっているSPARCチップをごく自然に使ってしまいます。

そのうち，パソコンでもRISCプロセッサは採用されて，

「RISCパワー全開！

　　　スピードアップ5倍（当社比）」

などという宣伝をするようになると思います。実際，いままで使っていたマシンが急に5倍速くなると世界はがらりと変わるものです。

ごく卑近な例を出しましょう。いまは，あるソフトハウスの技術力は，ワープロをバージョンアップしたときの，付加した機能あたりの速度の低下量によって，ある程度見積もることができると思われます。ここでいっているのはこういうことです。単に機能を増やしていくということは，ある程度の技術力があれば原理的に難しいことはない場合が多いのですが，種々の機能をどのようにうまく（速度が落ちないように）統括するかという基本的な設計というのは，よりマクロな視野が必要であり，ここのあたりがそのソフトウェアハウスの技術力を如実に物語るという意味です。

もし，急に5倍速くなったら，逆に速すぎるということになり，パラメータを入れて遅らせなくてはならなくなることも少なからず生ずるでしょう。したがって上で述べたような判断というのはあまり意味がなくなるわけです。ですから，革新的な速度向上がなされた後のしばらくの間は，ソフトウェアの速さとは単に時間待ちのパラメータということが多くなり，ソフトウェアの評価基準がより内容的なことに変化するということになると思われるのです。

RISCの次はVLIWの大合唱？

RISC一辺倒の様相を見せているワークステーション市場ですが，次の波は何かといえば，それはやはりVLIWということなのでしょう。VLIWはVery Long Instruction Word（とても長い命令語幅）の頭文字をとった言葉で，そのもつ意味自体は説明するまでもないでしょう。命令の1ワードが，16ビットや32ビットではなく，数百あるいは1,000ビットを越えるという想像を絶するような命令幅をもつアーキテクチャのことです。

VLIWは素晴らしいということを最初に知ったのは，現在は日本IBM東京基礎研究所の所長をされている鈴木則久先生の特別講義のなかででした。あまりにも突拍子もないというか，複雑な方式のような気がしたので，講義終了後に思わず，「そんなに複雑なソフトウェアとハードウェアが必要なのならば，計算モデルのすっきりしたデータフローマシンのほうがましではないか？」などという質問をしてしまいました。それに対する明確な答えというものは，記憶していないのですが，まあそのころデータフローマシンの勉強を始めたころでした（しかもデータフローマシンのもつ問題点まで理解していなかった）ので，思わずそんなことをいったのでしょう。

このVLIWアーキテクチャにおける1命令はいままでのプロセッサの命令とはまったく意味合いが異なります。ひとつのバカ長い命令は，いくつもの独立したオペレーション（操作）が寄せ集まったものです。たとえば，あるひとつの命令といっても，それはレジスタ1とレジスタ2を掛ける操作やレジスタ3とレジスタ4を足す操作やあるメモリ番地からデータをレジスタ5にもってくる操作などを自由に並べたものになっているのです（図1）。

もうおわかりだと思いますが，要するにVLIWは並列計算をひとつのプロセッサ内で実現しようとする，ひとつの並列実行方式なのです。そして，ひとつの命令の中に並べられた複数の操作はまったく同時に行われます。

でもまあ，いま説明しただけでは，どこがすごいのかわからなくて当然です。VLIWを次のトレンドの主役たらしめることができるのは，その裏にある「トレーススケジューリング」というソフトウェア上の大きな収穫があってこそなのですから。

CPUのビット数の意味

CPUのビット数といっても，目を付ける位置によって意味がまったく異なってきます。ちょっとVLIWそのものからはずれることになりますが，計算機にとってビット数が意味することは大事なことですので，さらっと見ておくことにしましょう。

最近のBYTE誌にちょうど「32ビットで十分なのか？」という記事[1)]がありまして，ビット幅の意味づけをいくつかの観点から紹介していますので，その分類に従って簡

単に見ることにします。

1) プロセッサのデータ

プロセッサの扱うデータは，そのプロセッサで走るプログラミング言語のもつデータタイプをサポートすべきである。

2) プロセッサの命令

RISCにせよCISCにせよ，命令語幅は32ビットで十分であると考えられる。

3) プロセッサのアドレス

各プログラム(プロセス)のもつアドレス空間は増大化するが，アドレッシングの技法で当分は対応可能である。たとえば，2つのレジスタをつなげて間接アドレッシングを使う方法がある。

4) システムバンド幅

プロセッサの速度に応じて，外部のメモリなどに対する転送量を増やす必要がある。この要求は，階層的なメモリ構成（キャッシュ）の採用である程度抑えることができる。必要な1秒当たりの転送量Cは次の式で表される。

$$C = MIPS \times (SZI + FRW \times SZD)$$

MIPS：1秒当たりにプロセッサが実行する平均命令数

SZI：命令の平均バイト数

FRW：メモリをアクセスする割合

SZD：アクセスデータの平均バイト長

たとえば，22MIPSのプロセッサで，SZI=4.0，FRW=0.3，SZD=3.8とすると，Cは，113となります。1秒当たり1Mバイトを転送しなければならないわけです。このような意味でバス幅を広げる必要が出てくるわけです。

なおこの記事の中では，2)を見るとわかるように，命令幅はいまのままでよいとしています。VLIWにも少し触れていますが，プロセッサ市場に影響を与えるのには最低でも5年はかかる（まったくの疑問符！）とだけいって深入りは避けています。

トレーススケジューリング

プロセッサの中にある演算ユニットALUなどをいくつも並べてひとつの命令で同時に動かし，並列度を得るということです。もし，10個のユニットをいつも同時に動かすことができたのならば，プロセッサとしては10倍速くなったことを意味します。このような考え方は，革新的でもなんでもなくて，並列実行の方法としてはきわめて自然な考え方だと思います。

VLIWアーキテクチャマシンが実際に現れなかったのは，基本ブロックの壁，わかりやすくいうならば分岐命令の壁があったからなのです。プログラム中に現れる分岐命令の割合はふつう10%から30%ぐらいでしょう。つまり，平均して3から8命令ごとにひとつはジャンプ命令が含まれているわけです。

そしてジャンプ命令があると，VLIW的な命令の詰め込みはしにくくなるわけです。特に条件分岐の場合はいずれにせよ条件判断の結果を待つことが必要になるわけです。要するに，条件分岐命令がふつうのプログラムではしょっちゅうあるために，プログラム自体のもつ並列度というものが限られてしまい，いくら複雑なALUを用意しても，同時に実行できる操作などあまり期待できないとずっと考えられてきたわけなのです。

ここで，生まれた画期的な手法が，トレーススケジューリング[2]です。トレーススケジューリングのすごいところは，if文で，式を評価した結果が真になるか偽になるかで操作を分断することなく，真になるか偽になるかを推定して確率の高いほうのパスを次々と選んだあと，そのパスの操作群をVLIW命令としてつなぎ合わせて最適化スケジューリングを行うというものです。

このスケジューリングでは，無数のブロック（条件分岐などで区切られる命令の固まり）の間を操作が移動することになります。ただし，期待していた真偽が逆に出た場合のために補償コードというものも埋め込む必要があります。

とにかく，この複雑なスケジューリングは，最近のソフトウェア技術の大きな収穫といえます。

VLIWはどこまで並列処理？

パソコンにはRISCが採用されるだろう，VLIW方式のマイクロプロセッサがいまに普及するだろうなどというような予測は，いうだけならば，きわめて簡単にいうことができます。大学などの研究レベル，メーカーの研究レベル，プロセッサのレベル，パソコンのレベルなどがあり，どんどん主力が変わっていくのですが，ただしそれらのレベルが伝搬するのには時間差がかなりあるからです（最初の2つはちょっと微妙ですが）。

VLIWの次の次の次ぐらいを目指している身分として，ここではVLIWのもつマイナス面をいくつか指摘して，次にたぶん流行するであろうVLIWに関する話をまとめることにしましょう。

VLIW思想の欠点その1

逐次型言語を最初からターゲットとしているため，シミュレーションで得た100近い並列度が現実にはほとんど得られない。

VLIW思想の欠点その2

トレースが当たっているうちは最適だが，だめな場合にはつけがくる（五分五分という条件分岐は苦手である）。

VLIWアーキテクチャにきわめて忠実なコンピュータ（1024ビットというとてつもない命令幅をもつ）がアメリカのマルチフローコンピュータ社から売られています。一度使ってみたいなあといつも思っていますけれどそのチャンスがなぜかありません。億単位のお金の使える人はぜひ買ってみてください。

図1 VLIWプログラムの例

a)
```
C=A+B
K=I*J
L=M-K
Q=C/K
```

b)
c)

LoadStore	IMul	LoadStor	Add	IMul
LOAD A	*	LOAD B	*	*
LOAD I	*	LOAD J	*	*
LOAD M	K=I*J	STORE C	*	*
*	*	STORE K	L=M-K	Q=C/K
STORE L	*	STORE Q	*	*

（*印はNOP）

a)のようなプログラムは，ふつうb)のような機械語に表すことができるが，VLIW計算機ではc)のような機械語になる。横1列がひとつの機械語命令であり，同時に実行されるので，たったの5ステップで実行されることになる。

参考文献

1) Steve Krueger, "Are 32 BIts Enough?", Byte, Vol. 14, No. 12, pp. 299-305 (1989,11).

2) J. R. Ells, "Bulldog: A Compiler for VLIW Architectures", The MIT Press (1986).

第43回

猫とコンピュータ

夢をセールス

Takazawa Kyoko
高沢 恭子

木枯らしの吹きぬける季節だというのに，いつもアクティブなキョウコさん。トークサロンで講演をしたり，パソ通仲間たちとイベントをこなしたりと，あいかわらず忙しい毎日を送っているようです。

なんだかとってもあったかい11月の荒川土手は，お散歩の人もたくさんいる。小学1年生のアッ（アキコ）ちゃんは，段ボールをひきずって，またエッサエッサと下から登ってきた。てっぺんにいるパパのそばまできて見上げる目が，思いっきり丸く大きく輝いている。いま2人は，おシリに段ボールをあてがって，土手すべりに励んでいるのだ。こんなにやりがいのあるマジメな遊びを理解できる大人は，パパのほかにはそんなに何人もいないのだから，アッちゃんは尊敬と満足でハズんでいる。この間はアッちゃんの自転車の練習なのに，マクドナルドのハンバーガーと，どうしてかラジコンの自動車も持ってきたパパなのだ。下のグラウンドでは，ヒット乱発の草野球チームが沸き返っている。

パパはちゃんと土手すべりの仲間を務めながら，頭の中では大小，粗密のさまざまな計画を並べかえている。

本業の会社の仕事は，もちろんいつもメニューにあるが，それが隅のほうに押しやられたり，下のほうに潰されたり。まずはいちばん近いイベントである，荒川区から委託された「川の手セールスマン」としての仕事，『パソコン通信フェア』の開催がある。

それから来春のFBI（通信ネット）本部の新ビルへの移転と，ネット運営の新展望の数々。背中には，発足したばかりの全国BBS連絡協議会の常任幹事としての任務のあれこれや構想，etc & etc。初期化されたばかりの計画もあれば，二重，三重の奥深い階層までできた計画もある。

そのすきまに，アッちゃんとお兄ちゃんのこと，ママのこと，空手（有段者です）のこと，いろいろ詰め込んで，パパは端正な顔ですまして土手すべりに励んでいる。

パパは七色

11月3日，池袋サンシャインシティでの某大手メーカー主催による「パソコンフェア'89」の最終日。催しのひとつである女性向けのトークサロンに講演者として招かれた。1日は評論家の五代利矢子さん，2日はテクニカルライターの高橋慈子さん，そして3日は私。約1時間の話のあと，パソ通の女性ネットワーカーが，ユーザーとして体験談を語るという形式だ。会員1000人の無料BBS，わがFBI-NETからも，シスオペ夫人の「えいりあんさちこ」さんがお呼びを受けて，1日の五代さんのあとにキャリアのある女性ネットワーカーとして，パソ通の実際についていくつかのお話をした。

私のほうは，あらかじめの打ち合わせや進行台本をちょうだいしていたものの，当日にならなければどんな方たちが集まるのか詳細は不明ということで，準備にも限界があった。聴講はハガキでの申し込みによるもので，約80名の定員だそうだ。

当日は文化の日でもあり，サンシャインシティはたいへんな人出で，トークサロンのほうもまあまあの満席だった。パソコンに興味のある方たちであるのは当然だが，まだパソコンには触れてみたことがないという人も3分の1くらい，ほかの人もおおむねは初心者だ。

語った内容は，私が知る限りでのパソコン約10年の流れと，私が見つけた「自分なりのパソコン」について。そしてわずかばかりの体験的アドバイスと，初心者には収穫の多いパソ通へのいざない。司会者の要領を得た進行で，大きな失敗もなく持ち時間を消化した。

「レディスサロン」と名づけられた集まりに，たったひとり女装もせずに座っていてくれた男の人は，アッちゃんとママのサチコさんを伴ったFBIのシスオペ，nin隊長ことナカムラ（中村守利氏）さんだった。

休日のパパのフリをして，時間を目いっぱい生かした動きをする人。さり気ないサポートをしてくれながら，nin隊長はサンシャインシティ丸ごとぶんくらいの情報を持って帰るだろう。本業に身を隠して（?）誰とでも軽やかに，時にクールに接しながら，いつも企画と夢を抱えていて，やがてみんなに分けてくれる，そんな人だ。

たのまれちゃった

すこし前のこと，用事つづきのあとFBIに何日ぶりかのアクセスをしたら，どうやら隊長が深夜のテレビ番組に「川の手荒川セールスマン」のひとりとして，PR出演したらしい。デザイナーのKIKUさんの作品であるFBIのネームプレートを持って，11月26日の「パソ通フェア」についてのインフォメーションをしたのだそうだ。「ビデオ編集がマズかった」とか「隊長，あがってましたね」なんていう書き込みがイベントボードに連なっていた。

荒川区が'86年から始めた新しい町づくり「川の手キャンペーン」は，毎年意欲的なテーマのもとに積み重ねをつづけているようだ。最初の2年はポスターや作品展で「川の手」の名やイメージを浸透させ，'88年には「川の手探偵団」の募集で区外の人も交えて荒川の良さを発見してもらい，その活動とその後の「全国探偵団会議」の報告結果から，'89年の新たなテーマを見つけ出した。

そこで結論された，区のセールスポイント『人の住むまち』をコンセプトに，荒川区の良さを宣伝すること。その方法として荒川区を「外」に向かって売り込んでくれ

る「川の手セールスマン」を募集，「セールスマン」各自にひとつずつのイベントを主催してもらうということだった。

自分たちの町の再発見や新しい「ふるさとづくり」は各地でも盛んだが，内外の住民をスタッフに巻き込んで意志的な参加をさせてしまう荒川区のキャンペーンは，企画がそのまま成果の一部にもなる立体的なアイデアだと思う。

「セールスマン」は現在，過去，将来の荒川区住民で，これならば私に聞いて，という得意分野を持った人。PRの方法とイベントの企画案の提出によって応募できる。

ところが，得意分野を持つ人で，イベントの開催をできる人の応募は，区がはじめ予想していた数になかなか達しなかった。荒川区民であるFBIのnin隊長は，すでに実績を知る区役所企画課内の探偵団事務局のカワクダ氏から，ぜひセールスマンとして応募のうえ，「パソコン通信」をテーマにイベントをお願いできないかと打診された。

「合格」した隊長は，さっそくセールスマンとしてひと肌ぬごうということになったわけだ。

区民に通信初体験を

たのもしいセールスマンたちの顔写真入りポスターは，JR，私鉄各駅に掲示され，イベントもつぎつぎと実施されていった。

運転手ひとりつきの都電7000型車両の重さを，何人の力で引けるかというクイズつきの「都電とつなひき」。「星の王子さま」とキャンペーン映画「隅田川ファンタジー」を上映する「納涼映画会」。お寺の本堂を会場にして，クラシックと童謡をプロの演奏で聴かせてくれる「さいこうじ（西光寺）こんさあと」。台本なし，自由参加で，素人役者が自分の考えや夢をお芝居で演じる「ARAKAWA演劇サーカス」。早稲田～三ノ輪間を，音楽と食事を楽しみながら優雅な気分にひたる「オリエント都電特別列車」。そんななかでも，nin隊長の担当する「パソコン通信フェア」は，区では初めての試みである。

いま全体的にパソコンへの関心が上昇しているなかで，特に「通信」への興味が高まってきている。テレビドラマなどの影響も大きく，女性の関心も強いそうだ。文化の日のトークサロンの折，「パソ通をやってみたい人」という司会者の問いに，7割近い人が手を挙げたのを見てもうなずける。これはなかなかタイムリーで，効果的なイベントになりそうだ。

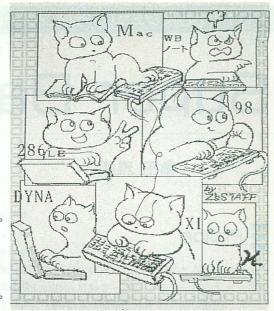
11月26日，会場は町屋文化会館1F多目的ホール。FBI-NET全体が隊長をバックアップする。会場を3つのセクションに分け，ひとつはBM（通信ホストプログラム，ビッグモデル）シミュレーション，ひとつはセミナー，もうひとつは，星占い，電子駄菓子屋，オセロゲーム，休憩場所にする。

セミナーの予定は，1時限めはMINE氏による「パソコン通信概論」。2時限めは隊長の「初心者のためのパソコン通信」で，これは実習。休憩のあと，PATA氏&BOB氏のコンビでのハイテンションジョッキー，本場アメリカの大手 BBS にアクセスして「英語でチャット」してみようというもの。

イベントボードの隊長の企画を見てIB PATA氏が「おお，英語チャットの練習しなくちゃだわ」なんて言ってる。

夢をおてつだい

通信シミュレーションは9回線で実施，うち1本は荒川NTTの協力で，電話回線によって外部との接続ができるようになっている。隊長は端末集めの協力を呼びかける。

「ラップトップ，もしくはワープロなど当日貸していただける方，またお願いします。いつものSSKさん，Impulseさん，あと純姫さんだっけ？　かな。9回線を予定していますので……2回線はX1グループにより予約であと7台です。星占い用に98がいるし，誰かMacでいらっしゃいましたね」「RS-232Cケーブル，セミナー用に1，BMシミュレーション用に9。特にリバースがいいんですけど，お持ちの方ご連絡ください」「ぶんさんへ。いつもハンダを持っての緊急メンテ担当の阿部さんが，アメリカへ出張になってしまいました。ぶんさんには，ハンダごてとテスター持参で，電気屋さんを当日お願いしたいと思います。もし，モジュラージャックを作る工具（名前忘れた）もあったらお願いします。部品はいくつか持っていきます。あの規模で，配線をやると1カ所は断線などがありますので，よろしく」「はい。りょーかい」とぶんさん。

「クロス1本確保しました。自作なんですが，テストの結果2400bpsで文字化けなし。3メートルあります」とながみね講師。「はい，了解。備品にはFDのシール等を使って名前がわかるようにしておいてくださいねぇ。帰ったら98が1台よけいにあったぁなんていうの困りますから」と隊長。

つづいて「286LEとクロス1本持っていきます」と純姫さん。「了解。よろしくお願いいたします。端末ソフトもね」と隊長。

「天婦羅★三杯酢さんへ。駄菓子屋さんの件です。セミナー開催中だけ，パソコンからの音楽を止められるようにプログラムしてください」

「CLOVIS君へ。会場警備（SWAT）もですけど，セミナー用CRTにセミナーをしていないあいだ電光掲示板みたいにメッセージを流しておきたいので，FDで一式作っておいてください。PDSでありましたよね。デカ文字の横スクロールするやつです。テキストは，当日書き込んでもOKです」

星占いは，広島のイマオカさんのプログラムを了解を得て拝借することになり，わが家にあったものをコピーして準備。これはプリントアウトのサービスつき。

オセロゲームは，ながみね講師のX1での作品。人工無能，愛称「美奈子」ちゃんも活躍するらしく，担当の夫も準備中だ。

夢をセールスするナカムラ隊長のもと，夢を手伝う喜びを込めて，みんなその日に備えている。

X1/turbo用シミュレーションゲーム

Super Battle

Kameda Masahiko 亀田 雅彦

大戦略ライクなゲーム性を盛り込んだメニュー選択方式のシミュレーションゲーム。軽快な思考ルーチンとスピーディな処理で白熱の戦闘フェイズが楽しめる。コンピュータ対プレイヤーだけでなく，人間同士，コンピュータ同士の対戦モードもあるぞ。

Prolog

中東のとある2カ国では，宗教対立に根ざすドロ沼の戦争が8年に渡って続けられていた。国連の調停作業も2年前から暗礁に乗り上げたままだ。お互いの港湾施設とことごとく破壊され，もはや石油の輸出もほとんどない状態だ。武器は，中古品の供与と借款によってまかなわれていたので，国の経済は両国とも崩壊寸前だったのである。

しかし，砂漠の民にとって，戒律の厳しい宗教は絶対不可侵なものとされてきた。戒律を破ることは，すなわち死を意味するものだった。したがって，両国民はそれぞれの唯一絶対神を信じ，その名を叫びながら戦場に散っていったのである。

そして忘れてはならない重要な要因がもうひとつある。それは，両大国の軍事的圧力である。一方はアメリカを中心とする西側世界とつながり，一方はソビエト・中国を中心とする東側と結びついていた。両国とも，表立って支援するようなことはなかったが，実戦に投入されている兵器からそのことはすぐに察せられるのだった。すでに両体制の代理戦争だったのである。

そんな折り，一方の国の指導者が突然死亡した。一部では暗殺説も流れたが，どうやら病死だったらしい。その国では，指導者の喪が明けしだい全面攻勢に出ると宣言。実際に相手の領土深くまで侵入し，首都に迫る勢いを見せた。相手側も国連を通じて声明を発表し，敵を侵略軍と位置づけた。お互いに手持ちの兵器は限られている。侵略軍対防衛軍の戦いは，短期決戦ですぐにも決着がつくだろう。君にはこの最後の攻防戦の司令官として，第一線で活躍してもらいたい。国の命運をかけて，君の健闘を祈る。

入力方法

オールBASICのプログラムなので，リスト1，リスト2の順で入力して，同じデバイス上にセーブしてほしい。その際，リスト1はなんでもいいが，リスト2は必ず“Battle.Bas”というファイル名にすること。リスト1，2とも，X1用BASIC(CZ-8FB01かCZ-8CB01)で入力・実行するようになっている。しかし，turboBASIC (CZ-8FB02) やZ-BASIC (CZ-8FB03) でも，1050行の前のほうの注釈「'」を取ることで動作可能である。また，CZ-8FB02/3で高解像度ディスプレイを使っている方は，1050行の前後両方の注釈を取ることで，高解像度で遊べる。

実行方法は，リスト1，2をセーブしたあとリスト1をRUNするだけである。PCGをセットして，自動的にリスト2を読み込みゲームが始まる。

PLAY

ゲームは2人対戦型で，侵略軍と防衛軍(先攻と後攻)に分かれて戦う。それぞれに6個の兵器と6個のイベントが配られるから，相手の兵器の種類を見ながら，自分の兵器とイベントをひとつずつ選ぶ。相手も選んでくるので，そこで兵器の基本攻撃値と防御値にイベントの修正値を加えた値で戦う。先攻兵器の攻撃と後攻兵器の反撃で1フェイズで，破壊された兵器はなくなり生き残ったものはまた使える。イベントは必ずひとつ消費する。このフェイズを6回やって，破壊された兵器の多いほうが負けとなり引き分けもある。6フェイズで1ターンで，4ターンまで行われ最終的な勝敗が決定する。2ターンずつ分け合った場合は防衛軍の勝ちとする。なお，1ターンごとに先攻と後攻は入れ替わる。1ターン生き残った兵器があれば次のターンに持ち越される。

実際のゲーム中では，すべてテンキーの2，8で選びリターンキーで決定するだけなので，プレイしてみれば要領はわかるはずだ。

起動するとまず，防衛軍と侵略軍の設定をする。それぞれの軍を，人間がやるかコンピュータがやるかを決定する。コンピュータ1と2の違いはあとで述べる。そのあとは，それぞれの軍の兵器生産パターンの決定だ。たとえば，空軍を選べば空の兵器しか配られない。バランスがよいのはすべての兵器が使えるパターンである。

ゲームが始まったら，先攻側(右)から兵器を選ぶ。兵器の下の青い数字は，左から対空攻撃値，対地攻撃値，防御値だ。後攻側(左)が選んだら表示が変わって，イベントを選ぶ。数字の意味は兵器と同じなのだが，特殊なのは「先制攻撃」だ。先攻側の特権で，先攻の攻撃が成功すれば後攻からの反撃ができなくなる。先攻が失敗すれば普通に反撃できる。なお，後攻が「先制攻撃」を選んでも何も起こらない。

そして双方ともイベントを選ぶと，右下に横倍角の数字が2つクルクル変わっていて，「A：6　D：8」とか出ているだろう。これらの数字で勝敗を決めるわけだが，プレイヤーはそれほど気にしなくてよい。リターンキーを押せば勝手に進行していくのである。ただ，いざというときに知っていると気合いが入るだろうから，詳しい説明をあとで行う。画面中央よりちょっと上の赤い星印は，破壊した敵の兵器数であり，これの多いほうがそのフェイズの勝利者である。

プログラム

プログラムのメインルーチンは1090～1190行で，ゲームの始めと終わり，それと4ターン分を処理している。そこで事実上の中核となっているのは，“fight”というラベルのサブルーチンで，6フェイズ分を担当している。

さらにそのなかでも重要なのが，“select”と“hantei”である。“select”は，兵器選択サブルーチンを呼ぶための上位ルーチンで，“select2”はイベント選択のそれである。人間か，コンピュータの思考ルーチンに行くのかの分かれ目がここだ。“hantei”は読んで字のごとく，兵器が破壊されたかどうかの判定ルーチンである。さっきはリターンキーを押せば勝手に進行すると書いたが，実は1890行のIF文のような処理をしている。リターンキーを押したときの倍角の2つの数字の合計がsiで，wdが修正防御値（兵器の基本防御値＋イベントの修正値），waが修正攻撃値。kw＝0が破壊されたということだから，wd－wa＋6がsiより小さいとき，破壊されたということである。これは攻撃側，反撃側ともに同じだ。ここぞというときは自分で計算してから，気合いを込めてリターンキーを押してほしい。

今度は，“think1～3”（略して1～3）の説明である。これらはいわゆる思考ルーチンで，1は初めに選ぶときのコンピュータ1であり，2は2であり，3はイベントを選ぶときのものだ。1のほとんどは乱数で，相手が空軍か陸軍のときだけ有効な兵器を探す。2は“ずる”をしていて，後攻のとき相手の手を盗み見て自分の手を決める。これで少しは強くなったかに思われたが，あまり成果がなかった。3では，まず先制攻撃を探し，次に相手に有効な攻撃イベントを探し，最後に防御イベントを探す順になっている。まだ甘いルーチンだとは思うが，おおまかなところは押さえてあるし，解析しやすいように簡単なところでやめておいた。もし気力のある方は考えてみてもらいたい。

最後にラベル“data”のデータ文は，使われる兵器のデータがそのまま入っている。改造する方は，西側東側・空と陸でそれぞれ分けてあるので，それを変えないように書き換えてほしい。イベントのほうは，先制攻撃以外は変更可能だ。このプログラムは，兵器の名前を変えるだけでSFからファンタジーにまでなってしまうのである。おまけにモジュール化も徹底してあるつもりだから，改造，移植する際にもわかりやすいと思う。

BASICで，「ちょっと暇つぶしできるような」をポリシーに作ったので，末永く遊んでやってほしい。

＊

人間対人間でプレイすると，相手の選んだ兵器がわかってしまう。絶対に見るなよ条約を結ぶなり，じっくり見たうえで戦う上級戦法をとるのもいいだろう。先攻後攻は入れ替わるので，どちらかが有利になることはない。

今回はもっとも効率的な運用を目指す思考型ゲームに，現代兵器を登場させ，思い入れを強める効果を狙った。某大戦略にもいえることだが，このゲーム性は突き詰めれば“じゃんけん”のそれと同じだ。ゲームとは，シンプルなゲーム性をどれだけ飾りつけられるかにかかっているのである。

変数表

変数	内容
nm$ (0,?)	兵器の名前
nm$ (1,?)	イベントの名前
at (0,?,x)	兵器の基本値
at (1,?,x)	イベントの修正値
	(x＝0～2 対空，対地，防御)
win (?)	ターンの勝ち数
rd (?,?)	ランダムパターン
tn (0-1)	人間かコンピュータ
tn (2-3)	兵器生産パターン
ft (0,?)	後攻の手持ち兵器
ft (1,?)	先攻の手持ち兵器
iv (0,?)	後攻の手持ちイベント
iv (1,?)	先攻の手持ちイベント
s	0が後攻 1が先攻
gm	現在のフェイズ
da	攻撃側兵器番号
dd	防御側兵器番号
ia	攻撃側イベント番号
id	防御側イベント番号

リスト1

```
1000 '
1010 ' Super Battle      list 1
1020 '
1030 CLS:DEFINT a-z
1040 LOCATE 10,9:PRINT "PCG SETTING...":FOR i=&H20 TO &HDF
1050 LOCATE 24,9:PRINT i:a$=LEFT$(CGPAT$(i),8):b$="":FOR j=1 TO 8
1060 a=ASC(MID$(a$,j,1)):b=a OR a*2:WHILE b>255:b=b-128:WEND
1070 b$=b$+CHR$(b):NEXT:DEFCHR$(i)=b$+b$+b$:NEXT
1080 b$=HEXCHR$("1054387C38541000"):DEFCHR$( 42)=b$+b$+b$
1090 b$=HEXCHR$("000000ACAC0C3800"):DEFCHR$(175)=b$+b$+b$
1100 b$=HEXCHR$("58FC585818306000"):DEFCHR$(187)=b$+b$+b$
1110 b$=HEXCHR$("00ACACAC0C183000"):DEFCHR$(194)=b$+b$+b$
1120 b$=HEXCHR$("30FC3030F4F43000"):DEFCHR$(206)=b$+b$+b$
1130 b$=HEXCHR$("0030B0B0BCBCB800"):DEFCHR$(217)=b$+b$+b$
1140 b$=HEXCHR$("7050700000000000"):DEFCHR$(223)=b$+b$+b$
1150 RUN "battle.Bas"
```

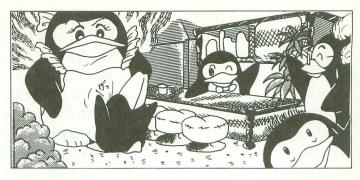
リスト2

```
1000 '
1010 ' Super Battle    list 2
1020 '                          '89 10  COPYRIGHT Kameda Masahiko
1030 '
1040 CLS4:WIDTH40:INIT:CLICK OFF:DEFINT a-z:CGEN 1
1050 'WIDTH40,25,0,1:KLIST 0:CONSOLE 0,25:KMODE 0:'CGEN 0:WIDTH4
0,25,0,2
1060 DIM nm$(1,20),at(1,20,2),kn$(1),pl$(2),xx(1),win(1),rd(2,6)
1070 DIM tn(3),ft(1,5),iv(1,6),ut(10)
1080 '
1090 GOSUB "start":GOSUB "init"
1100 FOR wpo=1 TO 4:LINE(1,1)-(18,5)," ",bf:LINE(21,1)-(38,5),"
",bf
1110 FOR s=0 TO 1:GOSUB "init2"
1120 ON tn(s+2)+1 GOSUB "shi","sha","sora","riku","baln"
1130 FOR w=0 TO 5:iv(s,w)=INT(RND*10+1):NEXT:NEXT
1140 GOSUB "fight"
1150 IF win0>win1 THEN m$=kn$(0)+"ﾉ ｶﾁ !!":win(0)=win(0)+1
1160 IF win0<win1 THEN m$=kn$(1)+"ﾉ ｶﾁ !!":win(1)=win(1)+1
1170 IF win0=win1 THEN m$="ﾋｷﾜｹ !!"
1180 GOSUB "printm":GOSUB "swap":PAUSE 30
1190 NEXT:GOSUB "ending":GOTO 1090
1200 '
1210 LABEL "init2"
1220 LOCATE xx(s),1:COLOR 3:PRINT "*";kn$(s):COLOR 7
1230 LOCATE xx(s),3:PRINT "ｶﾁ/ﾏｹ :";win(s);"/";win(1-s)
1240 LOCATE xx(s),4:PRINT "PLAYER:";pl$(tn(s))
1250 LOCATE xx(s)+1,5:COLOR 2:PRINT wpo;"turn":COLOR 7
1260 LOCATE 7,6:PRINT "        ":LOCATE 27,6:PRINT "        "
1270 CONSOLE 7,2:CLS:CONSOLE 0,25:RETURN
1280 '
1290 LABEL "swap"
1300 SWAP tn(0),tn(1):SWAP tn(2),tn(3):SWAP kn$(0),kn$(1)
1310 SWAP win(0),win(1):FOR i=0 TO 5:SWAP ft(0,i),ft(1,i):NEXT:R
ETURN
1320 '
1330 LABEL "shi":FOR i=0 TO 5:IF RND>.5 THEN w=1 ELSE w=11
1340 IF ft(s,i)=0 THEN ft(s,i)=INT(RND*5+w)
1350 NEXT:RETURN
1360 LABEL "sha":FOR i=0 TO 5:IF RND>.5 THEN w=6 ELSE w=16
1370 IF ft(s,i)=0 THEN ft(s,i)=INT(RND*5+w)
1380 NEXT:RETURN
1390 LABEL "sora":FOR i=0 TO 5:IF ft(s,i)=0 THEN ft(s,i)=INT(RND
*10+1 )
1400 NEXT:RETURN
1410 LABEL "riku":FOR i=0 TO 5:IF ft(s,i)=0 THEN ft(s,i)=INT(RND
*10+11)
1420 NEXT:RETURN
1430 LABEL "baln":FOR i=0 TO 5:IF ft(s,i)=0 THEN ft(s,i)=INT(RND
*20+1 )
1440 NEXT:RETURN
1450 '
1460 LABEL "fight"  'in tn(0-1),ft(?,?),iv(?,?)
1470 gm=0:win0=0:win1=0
1480 WHILE gm<6
```

```
1490 yu=10:yd=20:ys=2:GOSUB "select":GOSUB "select2"
1500 s=0:GOSUB "hantei":k0=kw
1510 IF k0<>0 THEN GOSUB "shippai":GOTO 1530 ELSE GOSUB "bakuha"
1520 COLOR 2:win1=win1+1:LOCATE 27,6:FOR i=0 TO win1-1:PRINT "*"
;:NEXT:COLOR 7
1530 IF ss=1 AND k0=0 THEN k1=ft(1,da):GOTO 1570
1540 s=1:SWAP da,dd:SWAP ia,id:GOSUB "hantei":k1=kw:SWAP da,dd:S
WAP ia,id
1550 IF k1<>0 THEN GOSUB "shippai":GOTO 1570 ELSE GOSUB "bakuha"
1560 COLOR 2:win0=win0+1:LOCATE  7,6:FOR i=0 TO win0-1:PRINT "*"
;:NEXT:COLOR 7
1570 ft(0,dd)=k0:ft(1,da)=k1:iv(0,id)=0:iv(1,ia)=0
1580 GOSUB "f2":gm=gm+1
1590 WEND:RETURN
1600 '
1610 LABEL "select" 'out da,dd
1620 FOR i=0 TO 5:ut(i)=ft(0,i):NEXT:xl=1 :s=0:GOSUB "utlist"
1630 FOR i=0 TO 5:ut(i)=ft(1,i):NEXT:xl=21:s=0:GOSUB "utlist"
1640 xl=21:s=1:ON tn(1)+1 GOSUB "cursor","think1","think2"
1650 da=d0:IF ft(1,da)=0 GOTO 1640
1660 xl=1 :s=0:ON tn(0)+1 GOSUB "cursor","think1","think2"
1670 dd=d0:IF ft(0,dd)=0 GOTO 1660
1680 GOSUB "f2":RETURN
1690 '
1700 LABEL "select2" 'out ia,id
1710 ia=0:id=0
1720 FOR i=0 TO 5:ut(i)=iv(0,i):NEXT:xl=1 :s=1:GOSUB "utlist"
1730 FOR i=0 TO 5:ut(i)=iv(1,i):NEXT:xl=21:s=1:GOSUB "utlist"
1740 xl=21:s=1:dt=dd:ON tn(1)+1 GOSUB "cursor","think3","think3"
1750 ia=d0:IF iv(1,ia)=0 GOTO 1740
1760 IF iv(1,ia)=5 THEN ss=1 ELSE ss=0
1770 LOCATE 21,yu+ia*2:CREV 1:PRINT SCRN$(21,yu+ia*2,18);:CREV 0
1780 xl=1 :s=0:dt=da:ON tn(0)+1 GOSUB "cursor","think3","think3"
1790 id=d0:IF iv(0,id)=0 GOTO 1780
1800 LOCATE  1,yu+id*2:CREV 1:PRINT SCRN$( 1,yu+id*2,18);:CREV 0
1810 RETURN
1820 '
1830 LABEL "hantei" 'in s  out kw
1840 GOSUB "f3":IF ft(s,dd)>10 THEN w=1 ELSE w=0
1850 wa=at(0,ft(1-s,da),w)+at(1,iv(1-s,ia),w)
1860 wd=at(0,ft(s,dd),2)+at(1,iv(s,id),2)
1870 LOCATE 29,24:PRINT "           ";
1880 LOCATE 29,24:PRINT "D";wd;"A";wa;
1890 GOSUB "sicoro":IF (wd-wa)+6<si THEN kw=0 ELSE kw=ft(s,dd)
1900 RETURN
1910 '*********    Sub-routine    ***********
1920 '
1930 LABEL "cursor" 'in xl,yu,yd,ys  out d0
1940 d0=0:yy=yu:m$="select !!":GOSUB "printm"
1950 LOCATE xl,yy:CREV 1:PRINT SCRN$(xl,yy,18);:CREV 0:KEY0,""
1960 w$=INKEY$:IF w$="" GOTO 1960
1970 SOUND 0,0:SOUND 1,15:SOUND 2,0:SOUND 3,15:SOUND 4,0:SOUND 5
,15
1980 SOUND 8,16:SOUND 9,16:SOUND 10,16:SOUND 11,0:SOUND 12,15
1990 SOUND 13,0:SOUND 6,0:SOUND 7,&B111000
2000 LOCATE xl,yy:PRINT SCRN$(xl,yy,18);:IF w$=CHR$(13) RETURN
2010 IF w$="8" AND yy>yu THEN yy=yy-ys:d0=d0-1
2020 IF w$="2" AND yy<yd THEN yy=yy+ys:d0=d0+1
2030 GOTO 1950
2040 '
2050 LABEL "random" 'out d0
2060 d0=INT(RND*6):RETURN
2070 '
2080 LABEL "think1"
2090 IF tn(3-s)=2 THEN w=0 ELSE IF tn(3-s)=3 THEN w=1 ELSE GOTO
2120
2100 ww=0:wr=INT(RND*3):WHILE ww<6 AND at(0,ft(s,rd(wr,ww)),w)<4
:ww=ww+1:WEND
2110 IF ww<6 THEN d0=rd(wr,ww):GOTO 2130
2120 GOSUB "random":w=255:IF ft(s,d0)=0 GOTO 2120
2130 RETURN
2140 LABEL "think2"
2150 GOSUB "think1":IF w<>255 OR s=1 GOTO 2200
2160 IF ft(1-s,da)>10 THEN w=1 ELSE w=0
2170 ww=0:wr=INT(RND*3):WHILE ww<6 AND at(0,ft(s,rd(wr,ww)),w)<4
:ww=ww+1:WEND
2180 IF ww<6 THEN d0=rd(wr,ww):GOTO 2200
2190 GOSUB "random":IF ft(s,d0)=0 GOTO 2190
2200 RETURN
2210 LABEL "think3"
2220 d0=0:WHILE d0<6 AND iv(s,d0)<>5:d0=d0+1:WEND:IF d0<6 GOTO 2
270
2230 IF ft(1-s,dt)>10 THEN w=1 ELSE w=0
2240 d0=0:WHILE d0<6 AND at(1,iv(s,d0),w)<=0:d0=d0+1:WEND:IF d0<
6 GOTO 2270
2250 d0=0:WHILE d0<6 AND at(1,iv(s,d0),2)<=0:d0=d0+1:WEND:IF d0<
6 GOTO 2270
2260 GOSUB "random":IF iv(s,d0)=0 GOTO 2260
2270 RETURN
2280 '
2290 LABEL "f2"
2300 CONSOLE 7,2:CLS:CONSOLE 0,25:LOCATE 2,7:CSIZE 1:PRINT#0 nm$
(0,ft(0,dd))
2310 LOCATE 25,7:PRINT#0 nm$(0,ft(1,da)):CSIZE 0
2320 RETURN
2330 '
2340 LABEL "f3"
2350 m$=nm$(0,ft(1-s,da))+"ｶﾞ "+nm$(0,ft(s,dd))
2360 IF s=0 THEN m$=m$+"ｦ ｺｳｹﾞｷ !!" ELSE m$=m$+"ﾆ ﾊﾝｹﾞｷ !!"
2370 GOSUB "hassha":GOSUB "printm"
2380 RETURN
2390 '
2400 LABEL "printm" 'in m$
2410 CONSOLE 23,2,2,26:CLS:LOCATE 2,23:COLOR 6:PRINT m$;:CONSOLE
 0,25:COLOR 7
2420 RETURN
2430 '
2440 LABEL "sicoro" 'out si
2450 LINE (29,23)-(37,23),"■",bf:KEY0,""
2460 w0=INT(RND*6+1):w1=INT(RND*6+1)
2470 LOCATE 28,23:CREV1:CSIZE2:PRINT#0 w0;:LOCATE 32,23:PRINT#0
w1;
2480 IF INKEY$<>CHR$(13) GOTO 2460
2490 CREV0:CSIZE0:si=w0+w1:m$="":GOSUB "printm":RETURN
2500 '
2510 LABEL "utlist" 'in xl,yu,yd,ut(0-),s
2520 LINE (xl,yu)-(xl+17,yd+1)," ",bf:dw=0
2530 FOR ii=yu TO yd STEP 2:LOCATE xl,ii:PRINT "*";nm$(s,ut(dw))
;
2540 LOCATE xl+4,ii+1:COLOR 1:IF ut(dw)=0 GOTO 2560
2550 FOR jj=0 TO 2:PRINTUSING " ## ";at(s,ut(dw),jj);:NEXT
2560 COLOR 7:dw=dw+1:NEXT:RETURN
2570 '**********     Sound      **********
2580 '
2590 LABEL "hassha"
2600 FOR ii=0 TO 3
2610 SOUND 0,10:SOUND 1,5:SOUND 2,10:SOUND 3,4:SOUND 6,4
2620 SOUND 8,16:SOUND 9,16:SOUND 11,10:SOUND 12,40:SOUND 13,0
2630 SOUND 7,&B100100:FOR jj=0 TO 400:NEXT:NEXT:RETURN
2640 LABEL "bakuha"
2650 SOUND 0,10:SOUND 1,12:SOUND 2,10:SOUND 3,9:SOUND 4,10:SOUND
 5,6
2660 SOUND 6,10:SOUND 8,16:SOUND 9,16:SOUND 10,16
2670 SOUND 11,10:SOUND 12,50:SOUND 13,0
2680 SOUND 7,0:RETURN
2690 LABEL "shippai"
2700 SOUND 7,&B111111:PLAY "v15o3e0r0e3:v15o3e0r0e3":RETURN
2710 '***********    Begin&End     **********
2720 '
2730 LABEL "start"
2740 CLS:win(0)=0:win(1)=0:ii=0
2750 LOCATE 6,4:CSIZE 3:PRINT#0 "Super Battle":CSIZE 0
2760 LOCATE 30,24:PRINT "(C) KAME";
2770 LOCATE 1,10:COLOR 4:PRINT "ﾎﾞｳｴｲｸﾞﾝ":LOCATE 21,10:PRINT "ｼﾝ
ﾘｬｸｸﾞﾝ"
2780 COLOR 7:FOR i=1 TO 21 STEP 20:RESTORE "d0":FOR j=12 TO 16 S
TEP 2
2790 READ a$:LOCATE i,j:PRINT a$:NEXT:xl=i:yu=12:yd=16:ys=2
2800 GOSUB "cursor":tn(ii)=d0:ii=ii+1:NEXT:ii=2
2810 FOR i=1 TO 21 STEP 20:RESTORE "d1":FOR j=12 TO 20 STEP 2
2820 READ a$:LOCATE i,j:PRINT a$:NEXT:xl=i:yu=12:yd=20:ys=2
2830 GOSUB "cursor":tn(ii)=d0:ii=ii+1:NEXT:CLS
2840 RETURN
2850 '
2860 LABEL "ending"
2870 IF win(0)>win(1) THEN m$=kn$(0):w0=win(0):w1=win(1)
2880 IF win(0)<win(1) THEN m$=kn$(1):w0=win(1):w1=win(0)
2890 IF win(0)=win(1) THEN m$="ﾎﾞｳｴｲｸﾞﾝ":w0=win(0):w1=win(1)
2900 m$=m$+"ｶﾞ ｺﾉｸﾆｦ ｾｲｱﾂｼﾀ !!":CLS:CONSOLE 5,15,9,22
2910 LOCATE 9,5:PRINT w0;"ｼｮｳ";w1;"ﾊｲﾃﾞ ";m$
2920 PRINT:PRINT:PRINT "push [RETURN]":CONSOLE 0,25
2930 IF INKEY$<>CHR$(13) GOTO 2930 ELSE RETURN
2940 '
2950 LABEL "square" 'in xl,yu,xr,yd
2960 LINE(xl,yu)-(xr,yu),"─":LINE(xl,yu)-(xl,yd),"│"
2970 LINE(xl,yd)-(xr,yd),"─":LINE(xr,yu)-(xr,yd),"│"
2980 LOCATE xl,yu:PRINT "┌";:LOCATE xr,yu:PRINT "┐";
2990 LOCATE xl,yd:PRINT "└";:LOCATE xr,yd:PRINT "┘";
3000 RETURN
3010 '
3020 LABEL "init"
3030 RESTORE "data":COLOR 5
3040 FOR i=1 TO 20:READ nm$(0,i):FOR j=0 TO 2:READ at(0,i,j):NEX
T:NEXT
3050 FOR i=1 TO 10:READ nm$(1,i):FOR j=0 TO 2:READ at(1,i,j):NEX
T:NEXT
3060 xl=0 :yu=0:xr=19:yd=6:GOSUB "square":xl=0 :yu=9:xr=19:yd=22
:GOSUB "square"
3070 xl=20:yu=0:xr=39:yd=6:GOSUB "square":xl=20:yu=9:xr=39:yd=22
:GOSUB "square"
3080 LOCATE 6,6:PRINT "┤        ├":LOCATE 26,6:PRINT "┤        ├"
3090 xx(0)=1:xx(1)=21:kn$(0)="ﾎﾞｳｴｲｸﾞﾝ":kn$(1)="ｼﾝﾘｬｸｸﾞﾝ"
3100 RESTORE "d0":FOR i=0 TO 2:READ pl$(i):NEXT:COLOR 7
3110 RESTORE "r0":FOR i=0 TO 2:FOR j=0 TO 5:READ rd(i,j):NEXT:NE
XT
3120 FOR i=0 TO 1:FOR j=0 TO 5:ft(i,j)=0:NEXT:NEXT
3130 RETURN
3140 '
3150 LABEL "data"
3160 DATA F-14ﾄﾑｷｬｯﾄ,10, 1, 6, F-15ｲｰｸﾞﾙ,    9, 5, 5
3170 DATA F/A-18ﾎｰﾈｯﾄ,8, 4, 5, A-10,         2,10, 2
3180 DATA ﾋｭｲｺﾌﾞﾗ,     1, 9, 1, Mig31,      10, 0, 6
3190 DATA Mig29,       8, 0, 6, Mig23,       6, 7, 4
3200 DATA Su-24,       3, 8, 5, ﾊｲﾝﾄﾞ,       0, 9, 1
3210 '
3220 DATA ﾚｵﾊﾟﾙﾄﾞ2,    0,10, 6, ｹﾞﾊﾟﾙﾄ,      8, 5, 4
3230 DATA 74ｼｷ,        0, 8, 5, ﾛｰﾗﾝﾄﾞ2,    10, 0, 2
3240 DATA ﾗﾝﾎﾞｰ*,      8,10, 1, T-72,        0, 9, 5
3250 DATA SA4ｶﾞﾈﾌ,    10, 0, 2, SA8ｹﾞｯｺｰ,    9, 0, 3
3260 DATA BRDM-2,      1, 8, 2, ｼﾞﾝﾐﾝﾍｲ,     1, 3, 0
3270 '
3280 DATA ﾀｲｸｳﾐｻｲﾙ,    5, 0, 0, ﾀｲﾁﾐｻｲﾙ,     0, 3, 0
3290 DATA ﾄﾏﾎｰｸ,       3, 2, 0, ｺｳｾｲﾉｳﾚｰﾀﾞｰ, 2, 0, 1
3300 DATA ｾﾝｾｲｺｳｹﾞｷ*,  0, 0, 0, P-3C,        3, 0, 1
3310 DATA ｼﾞﾗｲ,        0, 5,-2, ｲｰｼﾞｽｶﾝ,     5, 0, 1
3320 DATA ｽｺｰﾙ,       -2,-5, 5, ｷﾘ,         -2,-3, 3
3330 '
3340 LABEL "d0":DATA Human,Computer1,Computer2
3350 LABEL "d1":DATA ﾆｼｶﾞﾜﾍｲｷ,ﾋｶﾞｼｶﾞﾜﾍｲｷ,ｸｳｸﾞﾝ    ,ﾘｸｸﾞﾝ,balance
3360 LABEL "r0":DATA 3,5,0,2,1,4 ,2,1,4,3,5,0, 4,2,1,5,0,3
```

マシン語カクテル in Z80's Bar

第7回――ドライブに連れてって(1)――

シナリオ：金子俊一
特別監修：浦川博之
イラスト：山田純二

嫉妬に狂うようこちゃんをなだめるべく始まったドライブゲーム講座。今月は手始めにBASICでアルゴリズムを解説してから，必要となるマシン語のIOCSルーチンを作ってみることになりました。はたして君は，ようこちゃんをドライブに連れて行けるか？

♪カランコロ～ン（ドアが開く音）

マスター（以下M）：いらっしゃい。

西川善司（以下善）：ハックショ～イ，いやーなんだかクシャミが止まらなくってね。メアリーの風邪でもうつったかなぁ？

長老（以下老）：おやおや，メアリーちゃんが風邪をひいたとはのう。

善：ほら，メアリーったらディズニーランドにマゼラトップ(注1)なんか着てくるもんだから。

源光（以下光）：タンクトップだろ！　まぁしかたがないですね。キャリフォルニアでは1年中暖かかったたらしいから。冬着をあまり持ってないんですよ，彼女は。

ようこ（以下Yo）：ふ～～～ん。ずいぶんとお詳しいのね！

光：えっ。そっそんなことは……。

Yo：よく言うわよ。だいたいね～……☆∞♂<♀〆……。

老：ワッワシはちと用事を思い出したのでな……またくるとしよう。（いそいそ）

善：ぼっぼくも善司ソフトのほうが忙しいから……。（こそこそ）

♪カランコロ～ン　バッタン

M：あらら。

Yo：§℃÷★だしぃ。光なんてどわいっキライよ！　ビェ～～ン。

光：よっようこさん，そんなこと言わずに，ねっ，そうだ，今度の休みはドライブに連れてってあげるから。

Yo：ヒック……ヒック。ほんとに？

光：ほんとですって。ねえマスター，気分転換になるやつ2つお願いね。

M：ほーい，気分転館(注2)。

光：だいじょーぶなんですか，この液体？

M：味以外は保証するよ。まあグイっと。

Yo：ゴクゴク，ふぅ～。そーいえばさぁ光る君ぃ，ゲームセンターとかでドライブゲームがあるでしょ。あの地面がパタパタって向かってくるのってどうやるの？

光：う～ん，初心者にはちょっと。

Yo：どーせ，どーせ初心者ですよ～。きっとわたしなんかよりメアリーのほうがメアリーのほうが……ビェ～～～ン。

光：わかった。わかりました。説明します。説明させていただきます。ね，ねっ。

M：ほい，気分転館。

Yo：ゴクゴク。

光：ほいじゃあね，カチャカチャ……。

表1　3D DRIVE GAME IOCS TABLE

サブルーチン名	サブルーチンの機能	破壊レジスタ
#SYSINIT	IOCSイニシャライズ。	AF,BC,HL
#STICK1	JOYSTICK1のスティックの状態を調べる。Aレジスタに01H～09Hの値が入る。方向はテンキーの1～9と同じ。	AF
#STRIG1	JOYSTICK1のトリガの状態を調べる。Aレジスタに01H～03Hの値が入る。第0ビットがトリガ1，第1ビットがトリガ2に対応している。	AF
#PALET	DEレジスタを用いる。パレットコードDをカラーコードEに換える。BASIC風ならば「PALET D,E」になる。	なし
#MSG	DEで始まる文字列を00Hがあるまで表示する。	BC
#PRINT	HLレジスタの内容の下3桁を16進数で表示する。	AF,BC,DE,HL
#PRTHX	Aレジスタの内容を16進数2桁で表示する。	AF,BC,DE,HL
#PRT	Aレジスタを直接テキストVRAMに送る。	AF,BC DE,HL
#WAIT	アドレス@WAITの中身をカウンタとしたウエイト。	なし

・PRINTルーチン関係ではアドレス@XYの中身を(X,Y)として，画面に表示します。
・LOCATEは直接@XYをいじる。
・基本的に改行などはしない。

やっぱり基本はBASIC

光：（リスト1）を見てごらん。

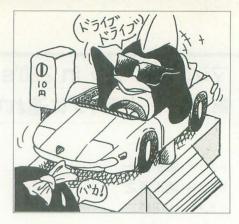
Yo：どれどれ，あっBASICじゃない。

光：うん，アルゴリズムの確認や研究には

図1　パレット機能のしくみ

BASICはもってこいだと思うよ。
Yo：ふ～ん。じゃあ実行してみよっと。あれー緑色の線が動いてるよ。
光：フフフフフ。
Yo：フが5つ。
光：いちいち数えなくていいの！
Yo：はーい。で，どーゆーしくみなの？
光：これはね，パレットを使っているんだ。
Yo：なにそれ？
光：ハードで瞬間的に色を変えるんだ。
Yo：カメレオン？
光：とは違うなぁ。カメレオンは本当に色を変えてるでしょ。パレット機能は本当は色を変えずに，見た目の色を変えるんだ。
Yo：ぜ～んぜんわかんない。
光：それじゃあこの図を見てごらん……カキカキ……（図1）。
Yo：なるほど，画面の色＝G-RAMの色ではないのね。
光：そういうこと。プログラムでは最初に全部の色をパレットで青色にして1色ずつ緑色に変えて青色に戻して……を繰り返してるんですよ。
Yo：でも，これじゃまるでどっかのビルのネオンサインみたい。
光：いまのはほんの小手調べ。ちゃんとしてないのは遠近感がないからだよ。
Yo：遠くの物は小さく見えて，近くの物は大きく見えるってやつね。
光：そう。それでは遠近感を出してみようかな。カチャカチャ……（リスト2）。

遠くの物はちいさくなあれ

Yo：今度はちゃんと動いてるみたいだけど，道もないし，カンジンの車もないわ。
光：草原ということにして，フロントガラスから見える風景ならば……。
Yo：いやっ！　道と車が欲しい。
光：ただ，そろそろスピード感がなくなってきちゃうんだよな，実際。
Yo：どうして？
光：だから，これからメインルーチンにゴチャゴチャ手を加えるんだけど，そのたびにスピードが少しずつ落ちていくんだ。俗にいう「処理が重くなる」ってやつさ。
Yo：シェイプアップする方法ないの？
光：『作ればやせる！　驚異の大減量プログラミング』ですか？
Yo：プログラミングでやせられるの？
光：そんなワケないでしょ。
Yo：なぁんだ。せっかくのチャンスだと思ったのにぃ。
光：そんなにやせたいの？
Yo：あのね，ダイエットして，1サイズ小さい服を着るの。クリスマスパーティでは髪を白いリボンで結んでね，やっぱりルージュは赤ねぇ。とっときのピンクのドレスに真っ白なヒール，ちょっと小さめのイヤリングとネックレスでキマリね。そうよ，今年こそみんなをあっと言わせるの。それからあの人に私の想いを打ち明けるの。彼ったら「僕も前から君のことが……」な～んちって。いや～困ったなぁ……。
（金縛りになっている光君）
M：ねぇ，プログラムはどうなったの？
Yo：そうよ。道と車。
光：もう，スピードがどうなってもいいんだね。じゃあ，カチャカチャ……，でーきたっと（リスト3）。
Yo：わーい，ドライブ，ドライブ。
光：ふう。
Yo：ちょっと，ほかに車は走ってないの？
光：あのねぇ，いまでも結構遅いでしょ。これ以上車を走らせたら渋滞5キロとかになっちゃうよ……。

マシン語するする

M：ねぇ光君，マシン語にはしないの？
光：もちろん直しますよ。
M：それじゃスピードには問題ないのね。
光：実はそのとおりなんです。
Yo：じゃあ最初からマシン語にすればよかったじゃない。
光：遅くても一応動くプログラムをBASICで組む。そのあとにマシン語に直すという手順はわりにオーソドックスなんだ。
Yo：どうして？
光：マシン語は一歩間違えるとすぐ暴走につながる。BASICならブレイクできるし，変数も保存されているからデバッグしやすいんだ。

――MASTER'S MEMO――

○DAA(Decimal Adjust Accumulator)は疑似的な10進数演算をする命令である。
たとえば，A＝59H，B＝36Hのとき，

```
    ADD A,B
```

では，A＝8FHだが，

```
    ADD A,B
    DAA
```

では，A＝95Hとなる。
見比べればわかるが，59＋36＝95という10進数の足し算のように16進数を扱えるのである。これを10進補正という。同様に，A＝95Hのとき，

```
    ADD A,$89
```

なら，95＋89＝184，よってA＝84H，CY＝1となる。
今回はスピード表示のところで使っているのだが，HLレジスタの下3桁を10進補正するようにプログラミングされている。参考にしてもらいたい。
○#WAITというサブルーチンでは8ビットのカウンタを使用しているが，もっとも一般的な16ビットのWAITルーチンも作ってみた（リスト6）。知らなかった人はぜひ解析していただきたい。
○ラベル名の話だが，今回のプログラムではIOCSサブルーチンの頭には#をつけ，ワークエリア関係には@がついている。こういった統一をしておくと大きなプログラムになったときにラベルを見ただけである程度の見当がつくので便利である。なお，ラベル名はそのルーチンが何をするか，何を表すのかをよく反映したものにするとよい。L370などというラベル名は対応するBASICリストがあるからよいものの，できれば避けるべきである。
○IOCSなどの仕様を決めるコツはメインプログラムのことをよく考えて，必要性のあるサブルーチンをリストアップし，使い方，注意点，レジスタ破壊などをメモしておく（プログラム中にも書いておくことがベスト）。1つひとつのサブルーチンをきっちり作り，デバッグもしっかりやっておく。この手のものを集めたライブラリがある人ほど開発効率がよい。

Yo：急がば回れってやつね。
光：そういうこと。
Yo：それじゃ，早くマシン語にしましょうよ。
光：え〜と，メインループのWHILE〜WENDの中にはどんな命令を使ったっけ？
Yo：う〜んと，PALETでしょ，FOR〜NEXTでしょ，STRIG，STICK，GOSUB〜RETURN，LOCATEとPRINTね。
光：GOSUBはCALL〜RETでそのまま代用できるでしょ。FOR〜NEXTもなんとかなるから，要するに必要なのはSTICK関係とPALET，PRINT関係だな。
Yo：そうなるかしら。
光：それじゃあ専用IOCSを作っちゃおう。
Yo：ICPO？　ゼニガタ警部でも呼ぶの？
光：IOCS。Input Output Control Systemの略だよ。つまりコンピュータの入出力をまとめて管理するプログラムなんだ。X1 turboのBIOSもこれに相当するね。さてと，カキカキ……。
Yo：それ，なにを書いてるの？
光：IOCSみたいな基本的なものは仕様をきっちり決めておくことがいちばん大切なんだ。ここでどの程度しっかり作るかによってバグの発生率がう〜んと変わってくるんだよ。
光：よーしっと，こんなもんか(表1)。
Yo：あとはプログラミングね。
光：まかしてちょーだい。カチャカチャ……。でーきったっと(リスト4+5)。じゃ，アセンブルして……。
Yo：ドライブ，ドライブ〜。???　ねえ光る君，確かにスピードは速くなったけど，ちょっとヘンだよ。
光：う〜ん。これはWAITルーチン（リスト6）を適当に作ったせいだなぁ。
Yo：なんとかしてよ〜。
光：自分でやりなさい，方法はいくらでもあるから。いい勉強になるよ。
Yo：ふ〜んだ。
光：さ〜てと，ということでそろそろ帰りますかな。
Yo：え〜っ。もう帰っちゃうの？
光：続きはまた今度。今月はページが足りないんだ。
Yo：そう。(さびしそうに)
光：来月また来ますよ。
♪カランコロ〜ン
Yo：あ〜あ，行っちゃった。
M：はい，気分転館。
Yo：もう，おなかいっぱいよ。

—つづく—

注1：懐かしいなぁ，ハモンさんが乗ってたっけ。「好きっだったよ，坊や」なんてセリフ残して……。
注2：ちゃんとした（?）飲み物。

リスト1　テストプログラム1

```
10 '  3D TEST 1
20 '
30 WIDTH 80 :INIT :CLR
40  '
50  ' * Draw Back
60  '
70 FOR I=1 TO 7 :L=i*3
80 FOR O=0 TO 199 STEP 21
90 LINE(0,O+L)-(639,O+L+2),PSET,I,BF
100 NEXT
110 NEXT
120  '
130  ' * Preparation
140  '
150 FOR J=2 TO 7
160 PALET J,1
170 NEXT
180  '
190  ' * Main Loop
200  '
210 WHILE 1
220 PALET 1,4 :PALET 7,1
230 FOR J=2 TO 7
240 PALET J,4 :PALET J-1,1
250 NEXT
260 WEND
```

リスト2　テストプログラム2

```
10 ' 3D TEST 2
20 '
30 WIDTH 80 :INIT :CLR
40  '
50  ' * Draw Ground
60  '
70  FOR O=1 TO 6 :L=(O-1)*7+L
80   FOR I=1 TO 7
90    LINE(0,79+L+I*O)-(639,79+L+I*O+O),PSET,I,BF
100   NEXT
110 NEXT
120  '
130  ' * Main Loop
140  '
150 WHILE 1
160  PALET 1,6 :PALET 7,4
170   FOR J=2 TO 7
180    PALET J,6 :PALET J-1,4
190   NEXT
200 WEND
```

リスト3　テストプログラム3

```
10 ' 3D TEST 3
20 '
30 WIDTH 40 :INIT :CLR :Q=5 :SP=150 :X=14 ' 3<=Q<=7
40 CONSOLE 0,25
50  '
60  ' * Draw Ground
70  '
80  FOR O=1 TO 10 :L=(O-1)*Q+L
90   FOR I=1 TO Q
100    LINE(0,79+L+I*O)-(319,79+L+I*O+O),PSET,I,BF
110   NEXT
120  NEXT
160 LINE (25,199)-(150,80),PSET,0
170 LINE (299,199)-(170,80),PSET,0
180 PAINT (160,80),0
190  '
200  ' * Draw Sky
210  '
220   COLOR 1
230  FOR I=1 TO 10
240   PRINT STRING$(40,"■");
250  NEXT
260   COLOR 7
270 LOCATE 10,0 :PRINT" SPEED=     Km "
280  '
290  ' * Main Loop
300  '
310 WHILE 1
320  FOR W=0 TO SP
330  NEXT
340 PALET 1,6 :PALET Q,4
350  FOR J=2 TO Q
360   PALET J,6 :PALET J-1,4
370      T=STRIG(1) :SP=SP+T*15
380      SP=SP+5+(SP>=250)*10-(SP<0)*15
390       LOCATE 16,0:PRINT 300-SP;
400       GOSUB 460
410       S=STICK (1) :X=X+(S=4)-(S=6)-(X<4)+(X>23)
420       GOSUB 510
430  NEXT
440 WEND
450 '
460 LOCATE X,20:PRINT"            ";
470 LOCATE X,21:PRINT"             ";
480 LOCATE X,22:PRINT"              ";
490 LOCATE X,23:PRINT"              ";
500  RETURN
510 LOCATE X,20:PRINT"   ／───＼  ";
520 LOCATE X,21:PRINT"  ▒─┌┼┐─▒ ";
530 LOCATE X,22:PRINT"███__███__███";
540 LOCATE X,23:PRINT"███‾*O+O*‾███";
550  RETURN
```

リスト4　テストプログラム4

```
10 ' 3D TEST 4
20 '
30 WIDTH 40 :INIT :CLR :POKE &HDF00,7,&H50,15 :CALL &HE000
40 '
50  ' * Draw Ground
```

```
60   '
70   FOR O=1 TO 6 :L=(O-1)*7+L
80     FOR I=1 TO 7
90       LINE(0,79+L+I*O)-(319,79+L+I*O+O),PSET,I,BF
100      NEXT
110    NEXT
120  LINE (25,199)-(150,80),PSET,0
130  LINE (299,199)-(170,80),PSET,0
140  PAINT (160,80),0
150    '
160    ' * Draw Sky
170    '
180  COLOR1
190    FOR I=1 TO 10
200      PRINT STRING$(40,"■");
210    NEXT
220      COLOR 7
230  LOCATE 10,0 :PRINT" SPEED=     Km "
240    '
250    ' * Main Loop
260    '
270  WHILE 1
280    CALL &HD000
290  WEND
```

リスト5 マシン語プログラム

```
0000                2 ; 3D DRIVE MAIN
0000                3 ;
0000                4
D000                5          ORG      $D000
D000                6
D000                7 MAIN
D000 11 06 01       8          LD       DE,$0106 ; { PALET 1,6
D003 CD 55 E1       9          CALL     #PALET   ;                 }
D006 3A 00 DF      10          LD       A,(@P)   ; {
D009 57            11          LD       D,A      ;   PALET Q,4
D00A 1E 04         12          LD       E,4      ;
D00C CD 55 E1      13          CALL     #PALET   ;                 }
D00F               14          ;
D00F 1E 01         15          LD       E,1      ;   {
D011 D5            16          PUSH     DE       ;
D012               17 MLOOP                      ;
D012 CD E3 E1      18          CALL     #WAIT
D015 D1            19          POP      DE       ;    FOR J=2
D016 1C            20          INC      E        ;
D017 7A            21          LD       A,D      ;
D018 BB            22          CP       E        ;       TO   Q
D019 D8            23          RET      C        ;
D01A D5            24          PUSH     DE       ;                 }
D01B               25          ;
D01B 53            26          LD       D,E      ; {
D01C 1E 06         27          LD       E,6      ;   PALET J,6
D01E CD 55 E1      28          CALL     #PALET   ;                 }
D021 15            29          DEC      D        ; {
D022 1E 04         30          LD       E,4      ;   PALET J-1,4
D024 CD 55 E1      31          CALL     #PALET   ;                 }
D027               32          ;
D027               33 L370
D027 CD 3F E1      34          CALL     #STRIG1  ; {
D02A 57            35          LD       D,A      ;
D02B 3A 01 DF      36          LD       A,(@WAIT);      T=STRIG(1)
D02E CB 1A         37          RR       D        ;
D030 D2 43 D0      38          JP       NC,BRAKE ;      SP=SP+T*15
D033 FE 10         39          CP       16       ;
D035 DA 3A D0      40          JP       C,STWAIT ;
D038 D6 0F         41          SUB      15       ;      SP=SP+5+(SP>250)*10
D03A               42 STWAIT                     ;
D03A 32 01 DF      43          LD       (@WAIT),A;        -(SP<0)*15
D03D CD E0 D0      44          CALL     SPDPRT   ;
D040 C3 4D D0      45          JP       RLCAR    ;      LOCATE 16,0
D043               46 BRAKE                      ;
D043 FE FA         47          CP       250      ;      PRINT 300-SP;
D045 D2 3A D0      48          JP       NC,STWAIT;
D048 C6 05         49          ADD      A,5      ;
D04A C3 3A D0      50          JP       STWAIT   ;                    }
D04D               51          ;
D04D               52 RLCAR
D04D CD 1C E1      53          CALL     #STICK1
D050 FE 05         54          CP       5
D052 CA 6E D0      55          JP       Z,CAR
D055 CD 7A D0      56          CALL     DELCAR
D058 21 02 DF      57          LD       HL,@CARX
D05B 7E            58          LD       A,(HL)
D05C DA 68 D0      59          JP       C,CARLEFT
D05F FE 18         60          CP       24
D061 D2 6E D0      61          JP       NC,CAR
D064 34            62          INC      (HL)
D065 C3 6E D0      63          JP       CAR
D068               64 CARLEFT
D068 FE 04         65          CP       4
D06A DA 6E D0      66          JP       C,CAR
D06D 35            67          DEC      (HL)
D06E               68 CAR
D06E 3A 02 DF      69          LD       A,(@CARX)
D071 32 03 DF      70          LD       (@XY),A
D074 CD B1 D0      71          CALL     PUTCAR
D077               72          ;
D077               73 NEXT
D077               74
D077 C3 12 D0      75          JP       MLOOP    ; NEXT
D07A               76          ;
D07A               77 DELCAR
D07A 3A 02 DF      78          LD       A,(@CARX)
D07D 32 03 DF      79          LD       (@XY),A
D080               80          ;
D080 F5            81          PUSH     AF
D081 21 03 DF      82          LD       HL,@XY
D084 7E            83          LD       A,(HL)
D085 23            84          INC      HL       ;HL=Y DATA ADDRESS
D086 36 14         85          LD       (HL),20
D088 11 05 DF      86          LD       DE,@SPS1
D08B CD 81 E1      87          CALL     #MSG
D08E 32 03 DF      88          LD       (@XY),A
D091 34            89          INC      (HL)
D092 11 10 DF      90          LD       DE,@SPS2
D095 CD 81 E1      91          CALL     #MSG
D098 32 03 DF      92          LD       (@XY),A
D09B 34            93          INC      (HL)
D09C 11 1C DF      94          LD       DE,@SPS3
D09F CD 81 E1      95          CALL     #MSG
D0A2 32 03 DF      96          LD       (@XY),A
D0A5 34            97          INC      (HL)
D0A6 11 1C DF      98          LD       DE,@SPS3
D0A9 CD 81 E1      99          CALL     #MSG
D0AC 32 03 DF     100          LD       (@XY),A
D0AF F1           101          POP      AF
D0B0 C9           102          RET
D0B1              103          ;
D0B1              104 PUTCAR
D0B1 21 03 DF     105          LD       HL,@XY
D0B4 7E           106          LD       A,(HL)
D0B5 23           107          INC      HL       ;HL=Y
D0B6 36 14        108          LD       (HL),20
D0B8 11 2A DF     109          LD       DE,@CAR1
D0BB CD 81 E1     110          CALL     #MSG
D0BE 32 03 DF     111          LD       (@XY),A
D0C1 34           112          INC      (HL)
D0C2 11 35 DF     113          LD       DE,@CAR2
D0C5 CD 81 E1     114          CALL     #MSG
D0C8 32 03 DF     115          LD       (@XY),A
D0CB 34           116          INC      (HL)
D0CC 11 41 DF     117          LD       DE,@CAR3
D0CF CD 81 E1     118          CALL     #MSG
D0D2 32 03 DF     119          LD       (@XY),A
D0D5 34           120          INC      (HL)
D0D6 11 4F DF     121          LD       DE,@CAR4
D0D9 CD 81 E1     122          CALL     #MSG
D0DC 32 03 DF     123          LD       (@XY),A
D0DF C9           124          RET
D0E0              125 SPDPRT
D0E0 21 11 00     126          LD       HL,$0011 ;
D0E3 22 03 DF     127          LD       (@XY),HL ;
D0E6 06 00        128          LD       B,$00
D0E8 4F           129          LD       C,A
D0E9 21 00 03     130          LD       HL,$300
D0EC ED 42        131          SBC      HL,BC
D0EE 7D           132          LD       A,L
D0EF 27           133          DAA
D0F0 6F           134          LD       L,A
D0F1 CD 97 E1     135          CALL     #PRINT   ;
D0F4              136
D0F4              137
DF00              138          ORG      $DF00
DF00              139 @P
DF00 00           140          DS       1          ; TOTAL PALET NO.
DF01              141 @WAIT
DF01 00           142          DS       1          ; SPD
DF02              143 @CARX
DF02 00           144          DS       1          ; X (FOR CAR)
DF03              145 @XY
DF03 00 00        146          DS       2          ; X ,Y
DF05              147 @SPS1
DF05 20 20 20     148          DM       "          "   ; SPS*10
DF08 20 20 20
DF0B 20 20 20
DF0E 20
DF0F 00           149          DB       $00
DF10              150 @SPS2
DF10 20 20 20     151          DM       "           "  ; SPS*11
DF13 20 20 20
DF16 20 20 20
DF19 20 20
DF1B 00           152          DB       $00
DF1C              153 @SPS3
DF1C 20 20 20     154          DM       "             " ; SPS*13
DF1F 20 20 20
DF22 20 20 20
DF25 20 20 20
DF28 20
DF29 00           155          DB       $00
DF2A              156 @CAR1
DF2A 20 20 20     157          DB       $20:$20:$20:$8F:$90:$90:$90
DF2D 8F 90 90
DF30 90
DF31 90 90 9F     158          DB       $90:$90:$9F
DF34 00           159          DB       $00
DF35              160 @CAR2
DF35 20 20 F0     161          DB       $20:$20:$F0:$F0:$90:$9E:$96
DF38 F0 90 9E
DF3B 96
DF3C 9B 90 F0     162          DB       $9B:$90:$F0:$F0
DF3F F0
DF40 00           163          DB       $00
DF41              164 @CAR3
DF41 87 87 87     165          DB       $87:$87:$87:$80:$80:$8D:$8D
DF44 80 80 8D
DF47 8D
DF48 8D 80 80     166          DB       $8D:$80:$80:$87:$87:$87
DF4B 87 87 87
DF4E 00           167          DB       $00
DF4F              168 @CAR4
DF4F 87 87 87     169          DB       $87:$87:$87:$7E:$2A:$E1:$2B
DF52 7E 2A E1
DF55 2B
DF56 E1 2A 7E     170          DB       $E1:$2A:$7E:$87:$87:$87
DF59 87 87 87
DF5C 00           171          DB       $00
DF5D              172
DF5D              173
DF5D              174 ;
DF5D              175 ; 3D DRIVE IOCS
DF5D              176 ;
DF5D              177
E000              178          ORG      $E000
E000              179
E000              180 #SYSINIT
E000 CD 00 E1     181          CALL     STKINIT
E003 CD 49 E1     182          CALL     PLTINIT
```

▶宮崎県産のピーマンで「Oh！宮崎」というのがあるのですが，やはり発売元は(株)日本ソフトバンクなのでしょうか（そんなわけないって）。　坊農　誠(18)福井県

```
E006 C9          183          RET
E007             184
E007             185 SPACE
E007 00 00 00    186          DS      $E100-SPACE
E0F1 00 00 00
E0F4 00 00 00
E0F7 00 00 00
E0FA 00 00 00
E0FD 00 00 00
E100             187 ;
E100             188 ; JOYSTICK ROUTINES
E100             189 ;
E100             190
E100             191 STKINIT
E100 01 00 1C    192          LD      BC,$1C00
E103 3E 07       193          LD      A,07
E105 ED 79       194          OUT     (C),A
E107 05          195          DEC     B
E108 ED 78       196          IN      A,(C)
E10A E6 3F       197          AND     $3F
E10C ED 79       198          OUT     (C),A
E10E C9          199          RET
E10F             200 STKIN
E10F C5          201          PUSH    BC
E110 01 00 1C    202          LD      BC,$1C00
E113 3E 0E       203          LD      A,$0E
E115 ED 79       204          OUT     (C),A
E117 05          205          DEC     B
E118 ED 78       206          IN      A,(C)
E11A C1          207          POP     BC
E11B C9          208          RET
E11C             209 #STICK1
E11C CD 0F E1    210          CALL    STKIN
E11F D5          211          PUSH    DE
E120 57          212          LD      D,A
E121 1E 03       213          LD      E,3
E123 3E 05       214          LD      A,5
E125             215          ;
E125 CB 1A       216          RR      D
E127 DA 2B E1    217          JP      C,STK1
E12A 83          218          ADD     A,E
E12B             219 STK1
E12B CB 1A       220          RR      D
E12D DA 31 E1    221          JP      C,STK2
E130 93          222          SUB     E
E131             223 STK2
E131 CB 1A       224          RR      D
E133 DA 37 E1    225          JP      C,STK3
E136 3D          226          DEC     A
E137             227 STK3
E137 CB 1A       228          RR      D
E139 DA 3D E1    229          JP      C,STK4
E13C 3C          230          INC     A
E13D             231 STK4
E13D D1          232          POP     DE
E13E C9          233          RET
E13F             234 #STRIG1
E13F CD 0F E1    235          CALL    STKIN
E142 F6 9F       236          OR      $9F
E144 2F          237          CPL
E145 07          238          RLCA
E146 07          239          RLCA
E147 07          240          RLCA
E148 C9          241          RET
E149             242 ;
E149             243 ; PALET D,E
E149             244 ;
E149             245
E149             246 PLTINIT
E149 21 7E E1    247          LD      HL,@BPLT
E14C 36 AA       248          LD      (HL),$AA
E14E 23          249          INC     HL
E14F 36 CC       250          LD      (HL),$CC
E151 23          251          INC     HL
E152 36 F0       252          LD      (HL),$F0
E154 C9          253          RET
E155             254 #PALET
E155 F5          255          PUSH    AF
E156 C5          256          PUSH    BC
E157 D5          257          PUSH    DE
E158 E5          258          PUSH    HL
E159 7A          259          LD      A,D
E15A 16 01       260          LD      D,$01
E15C B7          261          OR      A
E15D 28 05       262          JR      Z,PLT2
E15F             263 PLTLOOP
E15F CB 02       264          RLC     D
E161 3D          265          DEC     A
E162 20 FB       266          JR      NZ,PLTLOOP
E164             267 PLT2
E164 21 7E E1    268          LD      HL,@BPLT
E167 01 03 10    269          LD      BC,$1003
E16A             270 PLT3
E16A 7A          271          LD      A,D
E16B B6          272          OR      (HL)
E16C CB 0B       273          RRC     E
E16E 38 01       274          JR      C,PLT4
E170 AA          275          XOR     D
E171             276 PLT4
E171 77          277          LD      (HL),A
E172 ED 79       278          OUT     (C),A
E174 23          279          INC     HL
E175 04          280          INC     B
E176 0D          281          DEC     C
E177 20 F1       282          JR      NZ,PLT3
E179 E1          283          POP     HL
E17A D1          284          POP     DE
E17B C1          285          POP     BC
E17C F1          286          POP     AF
E17D C9          287          RET
E17E             288 @BPLT
E17E 00 00 00    289          DS      3
E181             290 ;
E181             291 ; PRINT ROUTINE 40*25 ONLY
E181             292 ;
E181             293
E181             294 #MSG
E181 F5          295          PUSH    AF
E182 D5          296          PUSH    DE
E183 E5          297          PUSH    HL
E184 1A          298          LD      A,(DE)
E185 FE 00       299          CP      $00
E187 CA 93 E1    300          JP      Z,MSG2
E18A D5          301          PUSH    DE
E18B CD BB E1    302          CALL    #PRT
E18E D1          303          POP     DE
E18F 13          304          INC     DE
E190 C3 84 E1    305          JP      #MSG+3
E193             306 MSG2
E193 E1          307          POP     HL
E194 D1          308          POP     DE
E195 F1          309          POP     AF
E196 C9          310          RET
E197             311 #PRINT
E197 7C          312          LD      A,H
E198 E5          313          PUSH    HL
E199 E6 0F       314          AND     $0F
E19B F6 30       315          OR      $30
E19D CD BB E1    316          CALL    #PRT
E1A0 E1          317          POP     HL
E1A1 7D          318          LD      A,L
E1A2             319 #PRTHX
E1A2 F5          320          PUSH    AF
E1A3 0F          321          RRCA
E1A4 0F          322          RRCA
E1A5 0F          323          RRCA
E1A6 0F          324          RRCA
E1A7 CD AB E1    325          CALL    PRTHX1
E1AA F1          326          POP     AF
E1AB             327 PRTHX1
E1AB CD B1 E1    328          CALL    ASC
E1AE CD BB E1    329          CALL    #PRT
E1B1             330 ASC
E1B1 E6 0F       331          AND     $0F
E1B3 F6 30       332          OR      $30
E1B5 FE 3A       333          CP      $3A
E1B7 D8          334          RET     C
E1B8 C6 07       335          ADD     A,7
E1BA C9          336          RET
E1BB             337 #PRT
E1BB CD C8 E1    338          CALL    ADRCAL
E1BE ED 79       339          OUT     (C),A
E1C0 3A 03 DF    340          LD      A,(@XY)
E1C3 3C          341          INC     A
E1C4 32 03 DF    342          LD      (@XY),A
E1C7 C9          343          RET
E1C8             344 ADRCAL
E1C8 D5          345          PUSH    DE
E1C9 E5          346          PUSH    HL
E1CA 21 00 00    347          LD      HL,$0000
E1CD 11 28 00    348          LD      DE,40
E1D0 ED 4B 03    349          LD      BC,(@XY)
E1D3 DF
E1D4 04          350          INC     B
E1D5             351 ADCL2
E1D5 19          352          ADD     HL,DE
E1D6 10 FD       353          DJNZ    ADCL2
E1D8             354
E1D8 59          355          LD      E,C
E1D9 19          356          ADD     HL,DE
E1DA 01 D8 2F    357          LD      BC,$3000-40
E1DD 09          358          ADD     HL,BC
E1DE 44 4D       359          LD      BC,HL
E1E0 E1          360          POP     HL
E1E1 D1          361          POP     DE
E1E2 C9          362          RET
E1E3             363 ;
E1E3             364 ; WAIT
E1E3             365 ;
E1E3             366
E1E3             367 #WAIT
E1E3 F5          368          PUSH    AF
E1E4 3A 01 DF    369          LD      A,(@WAIT)
E1E7             370 WAIT2
E1E7 CD F2 E1    371          CALL    WAIT3
E1EA 3D          372          DEC     A
E1EB FE 00       373          CP      0
E1ED C2 E7 E1    374          JP      NZ,WAIT2
E1F0             375          ;
E1F0 F1          376          POP     AF
E1F1 C9          377          RET
E1F2             378 WAIT3
E1F2 F5          379          PUSH    AF
E1F3 3E AA       380          LD      A,$AA   ; $AA is about.
E1F5             381 WAIT4
E1F5 3D          382          DEC     A
E1F6 FE 00       383          CP      0
E1F8 C2 F5 E1    384          JP      NZ,WAIT4
E1FB             385          ;
E1FB F1          386          POP     AF
E1FC C9          387          RET
E1FD             388
E1FD             389
```

リスト6 ウエイトルーチン

```
 1 ;
 2 ; WAIT ROUTINE
 3 ;
 4
 5 WAIT
 6          PUSH    AF
 7          PUSH    HL
 8          LD      HL,$XXXX    ; COUNTER
 9 WAIT2
10          DEC     HL
11          LD      A,H
12          CP      L
13          JP      NZ,WAIT2
14          ;
15          POP     HL
16          POP     AF
17          RET
```

▶わ〜っ，石を投げないで……。お，俺はX1turboからPC-8801FAに乗り換えたヒキョーものだよ!! でもこれはすべてX68000が高すぎるからいけないんだ!! FAなんて5万円で買ったんだゾ。……いいわけに聞こえる……できるだけ早いうちにXユーザーに戻りたい。
猪口 敏司(21)大阪府

特集2 Cプログラミング応用編

数式記述から3Dタートルまで

再帰大作戦

Tan Akihiko
丹 明彦

ワープロでは書きにくい数式もTeXなら美しく書けます。まず，再帰処理を使ってC言語でTeXもどきを作ってみましょう。一見複雑そうな処理もエレガントに記述できることがあります。いろいろな例題を通して再帰処理を考えます。

理系の学生なら，数学や物理のレポートを「ワープロで」書こうと試みた方は少なからずいると思うが，きっと一度や二度は挫折を味わったことだろう。コンピュータは数式を扱える機械である。計算するのはまあまあ得意技。しかし，体裁ということに関してはさっぱりである。これだけでは？？？の人のために，例を挙げよう。

1)　a^b

いうまでもなくaのb乗。BASICではa^b，FORTRANではa**b，今回の主役，Cにいたってはpow(a, b) などと関数でしか記述できない。しかし，まあここまでは，ワープロだったら1/4角文字を使って切り抜けられる(この原稿を書いているX68000のワープロでも，a^bは書けた)。それでは，ちと意地悪く，a^{b^c}はどうか？

2)

$$f(x)=\frac{1}{2}a_0+\sum_{n=1}^{\infty}(a_n\cos nx+b_n\sin nx)$$

これがなんの式かわかる人もわからない人も，試しにワープロで挑戦してみていただきたい。強引に全角・半角・1/4角などの文字と半分改行を駆使すれば，ひとつの式に3行くらい占領して，どうにか書くことはできるが，見栄えはきっといまいちであろう。ここでネックになっているのは，分数「1/2」と「Σ」である。それと，添え字「$_n$」も。

3)

$$a_n=\frac{1}{\pi}\int_{-\pi}^{\pi}f(x)\cos nx\ dx$$

$$b_n=\frac{1}{\pi}\int_{-\pi}^{\pi}f(x)\sin nx\ dx$$

今度は積分記号∫がネック。積分区間[$-\pi$, π] の表示が，ワープロではかなり苦しくなる。

図1　wp.xの出力例

$$f(x)=\frac{1}{2}a_0+\sum_{n=1}^{\infty}(a_n\ \cos\ nx+b_n\ \sin\ nx)$$

$$a_n=\frac{1}{\pi}\int_{-\pi}^{\pi}f(x)\cos\ nx\ dx$$

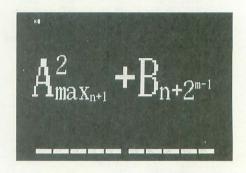
$$b_n=\frac{1}{\pi}\int_{-\pi}^{\pi}f(x)\sin\ nx\ dx$$

この，ワープロでは苦しい部分を専用のソフトウェアで見事に解決したのがTeXである。TeXは「テフ」または「テック」と読み，その筋では有名なDTP支援ツールである(正しくは“$\mathrm{T_EX}$”のように段違いに書くべきなのだが，ここでは“TeX”と書く)。ワープロの見苦しい印刷や，手書きの見づらい字から解放されるので，いまや理系の論文作成のために，素晴らしく強力な武器となっている。TeXは数式が「美しく」印刷できることが最大の特徴である。ワープロでも無理をすれば数式は書けるが，「どうにか書ける」という程度の品質の文字であり，TeXの文字とは次元が違うのだ。TeXで印刷された数式はまさに数学の教科書と見まがうばかりのクオリティ。コンピュータとレーザープリンタの組み合わせで(つまり印刷所を通さずに)，ここまでのことができる時代になったのだ。

TeXもどき

ということで，いきなりだがこのTeXのようなプログラムを作成してみたい。といっても，残念ながらΣや∫の話はただの余談。ここで扱おうというのはa^bとa_nだけ。以下，a^bのbのような文字を「上付き文字」，a_nのnは「下付き文字」と呼ぶ。情けない仕様だが，がっかりせずに先に進もう。名付けてTeXもどき。

とりあえず上付き文字について考えよう。上付き文字は，本体の文字より少し小さくなって，本体の右肩に付く。a^bでは，bがaより少し小さくなってaの右肩に付いている。a^{b^c}のcのような場合も同様で，cはbよりもさらに小さくなって，bの右肩に付く。

だいたいの構造がわかったところで，これをCのプログラムで実現することを考える。プリンタへの印刷は割愛して，グラフィック画面に表示することにする。これはもちろん，プリンタもなにも持っていない標準システムのユーザーを考慮してのことである，というのは嘘で，XCのsymbol関数が手軽に使えたからである。X-BASIC生まれのこの関数は，文字の大きさを自由に変えて表示できるので，TeXモドキにはちょうどいい

さて，a^{b^c}を表示する過程を追ってみたいのだが，その前にa^{b^c}をTeXではどう書くか見ていただこう。

```
$a^{b^c}$
```

注目したいのは，全角や1/4角などの特殊な文字が一切使われていないということ。TeXは通常の英文なども印刷できるようにするため，数式の部分を$……$でくくっているのだが，今回はこの部分はとりあえず目をつぶったフォーマットにしておく。つまりa^{b^c}はこう表現したい。

```
a^{b^c}
```

さあ，これを10秒ばかり眺めれば，TeXの上付き文字の規則についておおよそが理解できるであろう。

・数式の単位は1文字または{…}でくくられた数式(これを便宜的に「ブロック」と呼ぶ)。ブロックは，場合によってはネストすることもある(例：a^{b^{c^{d^e}}})。

・“^”の後ろの1文字または1ブロックは上付き文字になる。

ついでに下付き文字のこともいうと，

・“_”の後ろの1文字または1ブロックは下付き文字になる。

図2に間違いやすい例をいくつか載せておく。それを見てTeXモドキの文法を把握したら，プログラムの動作について考えよう。

a^{b^c}の「a」だけを指でちょっと隠してみてほしい。そこにはなにが見えるだろうか。

「b^c」が（少し小さくなっているけれども）見えるだろう。そうなのである。上付き文字というのは，文字を少し小さくして，普通の文字と同じように印刷しているにすぎない。aを表示したら，“^”の次のブロック{b^c}をaの右肩に上げ，そこから文字を小さくして“b^c”の表示にかかる。“b”を表示したところで，またも“^”が登場するので，その次の文字“c”を“b”の右肩に上げているのである。そしてさらに小さくなった“c”を表示しておしまい。

この解説は言葉よりも図で見たほうがずっとわかりやすいだろう（図3）。これを見ていると，なんだか似たような処理が積み重なっているような気はしないだろうか。するはずである。“^”が登場するたびに文字は小さくなるだけで，構文を解析して数式を表示する手順は文字がどの大きさのときでもまるっきり変わらない。

処理が繰り返しになっているので，ループを使ったらいいと考えた方もいらっしゃるだろう。しかし，ここで僕が気にいっている，わりとエレガントな方法を伝授したい。この記事の本題である「再帰」がそれである（前置きの長いことで）。

ここで，上付き文字と下付き文字をサポートした関数tex( )を考えよう。「文字列」はTeXフォーマットで書かれた数式で，倍率はいちばん大きな文字の拡大率。symbol関数とのからみもあるので，倍率には4とか8とか，切りのいい数で呼び出すとよかろう。以下は関数tex( )のアルゴリズムを日本語C（こんなもんあるんだろうか。昔，日本語BASICなどという大笑い言語もあったが）で書き下したもの。

```
tex（文字列，倍率）
{
  symbol（文字列のうち，この倍率で表示できる文字）;
  if（文字列中に^があったら）
    tex（上付き文字列，倍率/2）;
  if（文字列中に_があったら）
    tex（下付き文字列，倍率/2）;
}
```

tex( )の中でtex( )を呼んでいることに気づいただろうか。これを「再帰呼び出し（リカーシブコール）」という。自分自身（その関数自身）を呼び出しているところからこう呼ばれているのだ。

わざわざ再帰のような気持ち悪いものを使う理由があるのだろうかという疑問もわくかもしれない。僕が初めて再帰呼び出しに触れたときは少々気持ち悪いと思った。ふつうのBASICからX-BASICに移ってきたとき，自分を呼ぶという行為がGOTO，GOSUBに慣れた頭には奇異に映ったものだった。しかしいまは平気である。

a^{b^c}のような単調な数式ならば，ループでもなんとかなるのだろうが，ループでは融通がきくプログラムが作りにくいのも確かである。たとえば，「aのn₁乗」などと，上付き文字の中に下付き文字が紛れ込むようなややこしい場合を考えるとよい。ループだときっと汚いプログラムになってしまうであろう。それならいっそ再帰を使ったほうがプログラムもスッキリまとまるし，ずっと賢いやり方だと思う。

tex( )はごく簡単なサンプルだが，再帰のいくつかの美点が端的に現れている。

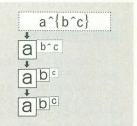
・プログラムが簡潔に，美しく書ける
・その割には難しい処理が実現できる
・プログラムの見通しがよくなる

再帰呼び出しは，関数のモジュール性を生かした，（おそらくは）高度なプログラミング技法である。tex( )は，数式の持っている再帰的な構造に着目して作ったプログラムといえる。リスト1がそうだが，使い方を説明しておこう。

まずコンパイルであるが，BASICライブラリを使っているので，その旨スイッチを指定しておくのを忘れずに（/W）。

図2　TeXもどきの入力と出力（間違いやすいもの）

入力		出力
a^b^c	→	a^{bc}
a^{b^c}	→	a^{b^c}
a^bc	→	a^bc
a^{bc}	→	a^{bc}
a^b_c	→	a^b_c
a^{b_c}	→	a^{b_c}

図3　TeXもどきにおける a^{b^c}の出力過程

リスト1

```
===============  tex.c  ==================
  1: /* TeXもどき */
  2:
  3: #include   <stdio.h>
  4: #include   <basic0.h>
  5: #include   <graph.h>
  6:
  7: #define BASE       0   /*おおもとモード*/
  8: #define SINGLE     1   /*1文字モード*/
  9: #define MULTI      2   /*ブロックモード*/
 10:
 11: #define BASE_X     0   /*表示開始位置*/
 12: #define BASE_Y   128
 13: #define BASE_MAG   4   /*表示開始倍率(8も試すとよろし)*/
 14:
 15: #define FONT       2   /*12×24ドットフォント*/
 16: #define SIZE      12   /*横方向ドット数*/
 17: #define COLOR      1   /*灰色*/
 18:
 19: int   tex( char*, int*, int, int, int, int );
 20:
 21: void  main()
 22: {
 23:    int   i=0;
 24:    char  formula[128];  /*数式を格納する文字列*/
 25:
 26:    screen( 2,0,1,1 );   /*768ドットモード*/
 27:
 28:    fgets( formula, 128, stdin );   /*標準入力から数式を読む*/
 29:    tex( formula, &i, BASE_X, BASE_Y, BASE_MAG, BASE );   /*第一段の呼び出し*/
 30:
 31:    return;
 32: }
 33:
 34: int   tex( s, i, x, y, mag, mode )
 35: char   *s;          /*文字列へのポインタ*/
 36: int   *i, x, y, mag, mode; /*(*i)は文字列の何文字目を読んでいるかを表す*/
 37: {                   /*x,yは表示位置、magは拡大率、modeは動作モード*/
 38:    char   c[2];  /*1文字symbolのための姑息な文字列*/
 39:    int   x1;      /*上付き/下付き文字混在表示用*/
 40:
 41:    c[1]=0;           /*文字列の終端コードは0*/
 42:    for (;;) {
 43:       switch ( c[0]=s[(*i)++] ) { /*文字列から1文字読む*/
 44:       case '^':            /*上付き文字*/
 45:          if ( s[*i]=='{' ) {        /*ブロックか?*/
 46:             (*i)++;                  /*かっこは読み飛ばし*/
 47:             x1=tex( s, i, x, y, mag/2, MULTI );  /*ブロックモードで再帰*/
 48:          } else {
 49:             x1=tex( s, i, x, y, mag/2, SINGLE ); /*1文字モードで再帰*/
 50:          }
 51:          if ( s[*i] != '_' ) x=x1;/*下付き文字混在なら表示位置は更新しない*/
 52:          break;
 53:       case '_':                  /*下付き文字:上付き文字とほぼ同じ*/
 54:          if ( s[*i]=='{' ) {
 55:             (*i)++;
 56:             x1=tex( s, i, x, y+mag*SIZE, mag/2, MULTI );
 57:          } else {
 58:             x1=tex( s, i, x, y+mag*SIZE, mag/2, SINGLE );
 59:          }
 60:          if ( s[*i] != '^' ) x=x1;
 61:          break;
 62:       case '¥n':
 63:          break;   /*以下のコメントをはずせば、改行にも対応する*/
 64: /*ここから       if ( mode!=BASE ) break;
 65:          x = BASE_X;
 66:          y += SIZE*BASE_MAG*3;
 67:          fgets( s, 128, stdin );
 68:          *i=0;
 69:          break;     ここまで*/
 70:       case '}':   /*ブロックの終わりは表示せずに戻る*/
 71:       case 0  :   /*文字列の終わりでも戻る*/
 72:          break;
 73:       default :   /*コントロール文字でない文字を表示する*/
 74:          symbol( x, y, c, mag, mag, FONT, COLOR, 0 );
 75:          x += mag*SIZE;   /*表示座標を更新する*/
 76:          break;
 77:       }  /*終了する文字ならループから抜ける*/
 78:       if ( c[0]=='}' || c[0]==0 || mode==SINGLE ) break;
 79:    }
 80:    return( x );   /*新しい表示座標を返す*/
 81: }
```

コマンドラインから,

A>tex

で起動し，入力待ちになる。適当な数式をタイプして最後にリターンキーを押せば，グラフィック画面にTeX風の数式が出てくる。たとえば，

a^{b^c}

とタイプすれば，

a^{b^c}

が画面に出てくることであろう。

ここで問題。複雑な数式をいちいちタイプするのは面倒だし，タイプミスも怖い。そんなときはどうすればいいだろう。そう，リダイレクションが使える。たとえば，

A>ed　abc. dat

として，

a^{b^c}

という内容のファイルabc. datを作る。そして，

A>tex　<　abc. dat

これでOK。いうまでもなくこれは標準入力を使ったトリックである。本筋とは関係ないが，参考までに。

さて，非常に心苦しいのであるが，簡単なプログラムだけに，いくつかの制限（もしくは欠陥）がある。小さくてわかりやすいプログラムのために実用性を犠牲にしたのだと善意に解釈してくれればうれしい。

・表現力が貧弱

上付き文字と下付き文字だけでは表現できる式もたかが知れている。もっともこれは仕様だからしようがない（へたなしゃれだこと）。

・半角文字しか受け付けない

大学数学/物理必須アイテムであるはずの∑も∫も∋も∇もαも使えない。ついでにカナも使えないのだが，こちらは実害はないであろう。どうしても使いたい方は，文字列を格納する配列にcharでなくunsigned char宣言を使うとよろしかろう。

・コントロール文字のチェックが甘い

改行コード（LF）は終了コードのひとつでもあるのでチェックしてあるが，そのほかはしていないので，思わぬ表示が出てきてびっくりするかもしれない。そのほかにも全般にエラーチェックは甘い。書式に合わない数式を入力した場合，まあ暴走はしないだろうが，きっと妙なことを始めるに違いない。

・手抜きである

上付き文字と下付き文字の混在は可能なように作ってあるが，プログラムが少し手抜きになっていて，2つのうち長くなるほうをあとに書かなくてはならない。これは次の2つの例を試してみるとよくわかることであろう。本来は同じ式のはずなのだが，結果は違ってしまう。というよりも，1番目の数式がおかしな表示をしてしまう。

x^{abc}_n+y

x_n^{abc}+y

再帰のあれこれ

さて，TeX（モドキ）をたたき台に上げて，再帰というものの考え方と有用性を力説したつもりである。再帰的な構造を持っている対象の多くは，ちょいちょいと再帰プログラミングしてやるだけであっけないほど簡単に処理できてしまう。そのいくつかを紹介してみよう。

・ハノイの塔

あまりにも有名。再帰プログラミングとは切っても切れない仲なのがこのハノイの塔。Pascalがパーソナルコンピュータ界に登場して，「これからは構造化言語の時代だ」と騒がれたころから，何回となくハノイの塔を見てきた。再帰呼び出しの使える言語なら，本当に簡単にできてしまう。ハノイの塔のなんたるかは，本誌1988年5月号で相馬氏が詳説されているのでそちらをご覧いただきたい。しかしアルゴリズムのキモの部分は「日本語C」で書きたいので書く。

```
hanoi（n枚の円盤をAからBへ動かす)
{
  hanoi（n－1枚をAからCへ動かす)；
  AからBへ，Aに残った1枚を動かす；
  hanoi（n－1枚をCからBへ動かす)；
}
```

う，美しい。総数2^n回に及ぶ膨大な数の手順は，再帰呼び出しが深くなるにつれてスタックに積まれていく。いわばスタックにシワ寄せがいっている，CPUが苦労するパターンではある。が，そのおかげでCのプログラム自体は非常に簡単にまとまっている。

・レイトレーシング

手前味噌な話で恐縮だが，レイトレーシングが屈折・反射を得意としているという事実は，再帰呼び出しをアルゴリズムに組み込んでいるということからきている。

```
ray（視点，視線ベクトル)
{
  視線と各プリミティブの交点を求める；
```

再帰と自動変数

Cの関数は（うまく書けば）モジュール化できる。中身を知らなくても，呼び出し方だけ知っていれば自由に使える独立したサブルーチンになり得るのである（ここがふつうのBASICのサブルーチンと大きく異なるところであろう)。関数の中で宣言した変数が外部とまったく独立しているので，「こちらを立てればあちらが立たず」式のバグの発生を抑えやすくなるのである。したがってデバッグもかなりしやすくなる。本文のtex( )の場合も基本的に内部だけ考えていればよい。次の段の再帰呼び出しの際に，拡大率を半分にしてやるだけで，実際に表示するグラフィック座標のことを神経質に考える必要がないのだ。

再帰呼び出しのためには，自動変数の存在が必要である。通常の変数はメモリのどこかに記憶領域をとって，その一定位置に格納される（Cでは静的変数と呼ばれる)。自動変数は関数を呼び出してその中で変数が宣言されると同時にスタック上に生成され，変数の値もスタックポインタ相対で参照される。関数の処理が終了して呼び出し元に帰るときに，自動変数はスタックから消えてなくなる。

tex( )を例に取ると，x, yなどは外部と独立しており，もし呼び出し元に同じ名前の変数xやyがあったとしても，縁もゆかりもない。再帰呼び出しでtex( )がtex( )を呼び出すような場合でさえも，呼び出し元のtex( )内のx, yと，呼び出し先のtex( )内のx, yはなんの関係もない。逆にいえば，こうやって変数の保護を保証しているから，関数が独立して動作でき，関数のモジュール化が可能になり，他人の書いた関数を，中身を少しも知らなくても，ブラックボックスとして，ライブラリとして，活用できるのである。自動変数は関数内のごく限られた領域でしか有効ではないので，変数の管理が楽になる。巨大なプログラムを書いたときに，この変数はどっかで使ってたような気が……といった悩みもなくなるし，それが原因で起こるバグも発生しなくなる。

いいことずくめではない。自動変数をスタック領域に確保する作業は関数を呼び出すたびに行われる。このオーバーヘッドのおかげで，スピード重視の計算屋さんからは「速度の低下を招く」として不評である。一応，弁護しておくならば，そんなときには静的変数の宣言をしておけば若干のスピードアップにはなる。

```
(auto) int　x, y;　/＊自動変数/
```

の代わりに,

```
static int x, y;　/＊静的変数＊/
```

なる宣言で，この変数はスタック上でなくメモリの定位置に確保されることになる。もちろんこの場合も関数外部との独立性は確保されている。しかし，再帰呼び出しの動作は保証されなくなってしまうので，できるだけ自動変数を使うべきであろう。わざわざautoという宣言をしなくても，ふつうに宣言した変数はすべて自動変数になる。

また，あえて独立していないほうがいい変数もある。グローバル変数と呼ばれている。いくつかの関数からも参照されるような変数は，main( )の前で宣言しておくことで，プログラムのどこからでも参照できるようになる。

再帰呼び出しはかなりループに展開できるといわれている。実際そうだと思う。しかし「バグの可能性はあるしプログラムも見にくくなるが高速」と「若干遅くてもエレガント」だったら，個人的には再帰のほうを選びたい。

交点がいちばん近いプリミティブについて：
if（散乱反射体）{
　表面色を計算する；
}
if（反射体）{
　反射ベクトルを計算する；
　ray（交点，反射ベクトル）；
}
if（透明体）{
　屈折ベクトルを計算する；
　ray（交点，屈折ベクトル）；
}
交点がなければ背景色を返す；
}

プリミティブと視線の交点が新しい視点となり，反射方向または屈折方向のベクトルが新しい視線となる。こうして再帰的に視線探索を行えば，反射/屈折をかなり忠実にシミュレートできることになる。

レイトレーシングは生成する画像の品質のわりに，原理的なアルゴリズムは意外とやさしい。原理的と断ったのは，もちろんこれだけでは遅いからで（再帰を使っているためではない，念のため），ボクセル分割などの高速化手法を導入するとアルゴリズムは複雑になっていく。マッピングなども難しい問題のひとつだが，これはレイトレーシングに固有な問題ではない。基本的にレイトレーシングは単純明快なアルゴリズム。これは再帰呼び出しの力によるところが大きい。再帰呼び出しを禁止しているFORTRANでレイトレーシングを書こうとすると，スタックをプログラマ自身が用意するハメに陥る。涙ぐましき努力といってもいい。やっぱりCが好き。

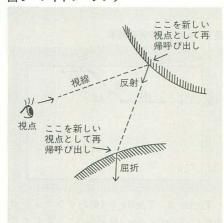
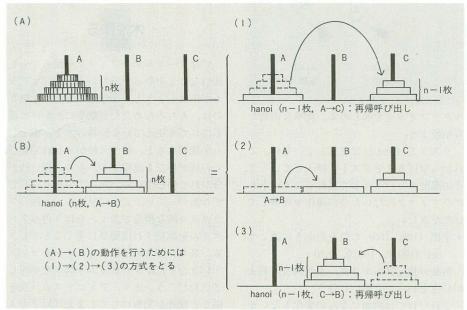
図4　ハノイの塔とその解法

Cによるタートルグラフィック

話が図形関係に集中して非常に心苦しいのだが，再帰呼び出しの例題その2として，「タートルグラフィックによるフラクタル図形」をとりあげる。これも12月号で泉氏が解説された題材であるが，ここでもう一度解説を試みよう。

タートルグラフィックは，実はCではなくLOGOなどの得意技である。タートル（turtle）は亀。いまでこそ画面上を動き回るものはねずみ（マウス）であるが，タートルグラフィックは，画面上を歩き回る1匹の亀になぞらえて命名されている。亀とはとどのつまりグラフィックカーソルの愛称である。

この亀さんの働きは実に単純である。できることといったら，向きを変えることと，直進することだけ。そして直進するときには，画面上に線が残る。いってみれば亀の足跡のようなものである。亀を制御する関数だけを集めてturtle.cというインクルードファイル（リスト2）に収めてある。

図5　レイトレーシング

リスト2

```
==================  turtle.c  ==================
 1: /*タートルグラフィック：基本サブルーチン*/
 2:
 3: #include   <math.h>
 4: #include   <graph.h>
 5:
 6: double   Turtle_x, Turtle_y, Turtle_arg;   /*グローバル変数：亀の状態を表す*/
 7: int   Turtle_color;
 8:
 9: void   set( double, double, double );      /*亀の状態を設定する*/
10: void   get( double*, double*, double* );   /*亀の状態を返す（保存するときに使う）*/
11: void   forward( double );         /*指定した距離だけ前進する*/
12: void   right( double );           /*指定した角度だけ回転する（右回りが正）*/
13: void   pencolor( int );           /*足跡の色を設定する*/
14:
15: void   set( x, y, arg )           /*亀の状態を設定する*/
16: double x, y, arg;                 /*引数は値*/
17: {
18:    Turtle_x=x;
19:    Turtle_y=y;
20:    Turtle_arg = arg/180.0*PI;   /*角度の単位をラジアンに変換する*/
21:    return;
22: }
23:
24: void    get( x, y, arg )          /*亀の状態を返す*/
25: double  *x, *y, *arg;             /*引数はポインタ（上と比較せよ）*/
26: {
27:    *x=Turtle_x;
28:    *y=Turtle_y;
29:    *arg=Turtle_arg*180.0/PI;    /*角度の単位を度に戻す*/
30:    return;
31: }
32:
33: void    forward( length )         /*指定した距離だけ前進する*/
34: double  length;
35: {
36:    double   x, y;
37:
38:    x=Turtle_x;                    /*今までいた位置*/
39:    y=Turtle_y;
40:    Turtle_x += cos( Turtle_arg )*length;   /*相対移動量を足して新しい位置を出す*/
41:    Turtle_y -= sin( Turtle_arg )*length;
42:    line( (int)x, (int)y, (int)Turtle_x, (int)Turtle_y, Turtle_color, 0xFFFF
);
43:
44:    return;
45: }
46:
47: void    right( arg )              /*指定した角度だけ回転する*/
48: double  arg;
49: {
50:    Turtle_arg -= arg/180.0*PI;  /*単位はラジアン*/
51:    return;
52: }
53:
54: void   pencolor( col )            /*足跡の色を設定する*/
55: int   col;
56: {
57:    Turtle_color=col;
58:    return;
59: }
```

リスト3

```
================== test.c ==================
 1: /* タートルグラフィックのテスト */
 2:
 3: #include   <basic0.h>
 4: #include   "turtle.c"
 5:
 6: void   main()
 7: {
 8:     int   n;
 9:     screen( 2,0,1,1 );
10:     pencolor( 1 );                  /*灰色*/
11:     set( 100.0, 100.0, 0.0 );   /*右向きに亀を置く*/
12:     /***ここに好きな文を入れてみよう***/
13:     return;
14: }
```

図6　turtle. c で定義されている関数

set (x, y, a)	カメを座標 (x, y) に，角度a°で置く[1]。
get (&x, &y, &a)	カメの状態をx, y, aに保存する。
right (Δa)	カメの向きを変える[2]。
forward (ℓ)	カメをℓだけ歩かせる。
pencolor (c)	足跡の色をcに変える。

x, y, a, Δa, ℓ はdoubleで
c はintで与えること。

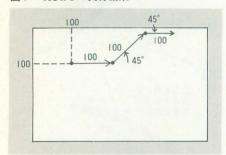
1)　角度

90°　a°　180°　0°　360°　270°

2)

Δa　Δa

Δa＞0　右回り　　Δa＜0　左回り

図7　test. c の実行結果

Turtle_x，Turtle_y (亀のグラフィック座標)，Turtle_arg (亀の向き)，そしてTurtle_color (足跡の色) はいわゆるグローバル変数になっている。だからmain( ) からでも値をいじるのは可能だが，それはしないというのがCプログラマの守るべきひとつの礼儀であろう。無理にいじらなくても，turtle. c内の関数で操作は十分できる。こうすることで初めて関数のモジュール化ができるのである。

たとえば足跡の色を変えたいときは，pencolor( ) を使わなくても，Turtle_color＝1で可能なのだが，インクルードファイルにした段階で，すでにプログラマが変数名Turtle_colorを「知らない」という前提のもとに置かれたことを意識しておこう。turtle.cに収めた関数を図6に示す。

これさえできれば，あとはmain( ) 中に亀の動き方を指示してやるだけで，タートルグラフィックが楽しめる。簡単な例題から始めよう。

リスト3 (test.c) は完結していない。これから挙げる文をリストに挿入していって，その都度コンパイルして実行し，タートルグラフィックのだいたいの操作を理解していただきたい。

・座標 (100, 100) に亀を右向きに置こう。

　set (100.0, 100.0, 0.0);

角度の指定は右が0度，上が90度。右上が45度といったぐあいに行う。

・これだけでは画面になんの変化もないから，亀を歩かせてみよう。

　forward (100.0);

画面に水平に線が引かれる。

・45度左折してみよう。そのあと前進。

　right (−45.0);

　forward (100.0);

right( ) は右折する関数。だから左折するには，角度に負数を指定すればよい。

・45度右折して前進する。

　right (45.0);

　forward (100.0);

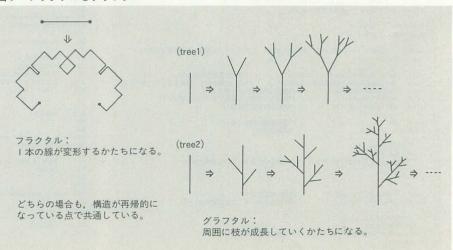
画面には図7のように折れ線ができているはず。

ここまででタートルグラフィックの使いやすさに気づかれたと思う。グラフィック座標を意識するのは初めだけで，あとは自動車を動かすような感覚で亀を動かすことができる。line( ) だと，始点と終点の座標を指定しなくてはならず，この計算がけっこう面倒になることがある。

木を作る

ではタートルグラフィックを使って少し面倒なことをやってみたい。木をプログラムで生成しようというのである。木というのは，もちろんあなたが散歩に出かければどこにでも生えている植物のこと。木をじっと観察すると，幹から枝が分かれて伸び，その枝が伸びる途中でも，もっと細い枝が分かれて伸びている。木はこの繰り返しでできあがっているように見える。一種のフラクタル的な構造であり，やはり再帰プログラムを駆使すれば簡単に書くことができる。名づけて木の生成シミュレーション。

いきなりだがプログラムリストを参照していただこう。まず，アルゴリズム。「枝を描く」機能を関数にしてしまえば以下のようになる。

図8　フラクタルとグラフタル

```
tree (サイズ)
{
  forward (幹または枝を描く);
  right (＋方向に角度をつける);
  tree (少し小さなサイズ);  /＊再帰＊/
  right (－方向に角度をつける);
  tree (少し小さなサイズ);  /＊再帰＊/
}
```

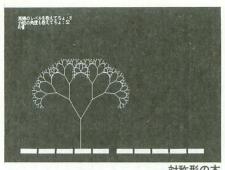
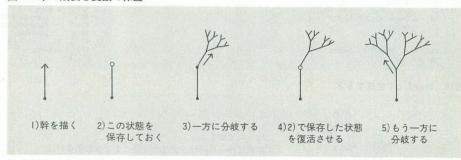
このようなプログラムで次々に2分割する枝を持つ木を描くことができる。

ただし，描画にはタートルグラフィックを用いるのだから，これだけではうまくいかない。なぜなら亀は基本的に1匹なので，同時に2方向に分岐する用途には向かないからだ。そこでソースリストのように，get( )でいったん現在の亀の状態を保存しておき，一方の枝を分岐させる（ひとつ目の再帰呼び出しをかける)。この枝を先端まで描き終わったら，再帰呼び出しから戻ってくるので，そのときに亀も呼び戻す。保存しておいた状態をset( )を用いて復活させ，もう一方の枝を分岐させる（2つ目の再帰呼び出しをかける）のである。図9をご覧いただきたい。

もうひとつ。再帰呼び出しのレベルが際限なく深くなるのは好ましくない。呼び出しの際にレベルを引数として与えるのは12月号のX-BASIC入門で作っていたコッホ曲線などと同じ。コッホ曲線では，もっとも深いレベルのところだけで線を引いていたが，木の場合は，どのレベルでも幹(または枝)を描き，もっとも深いレベルでは葉っぱを描くようにした。といっても，なんのことはない，色を変えているだけのこと。

これらも考慮すれば，tree( )は次のように書ける。リストのtree1.cの日本語版である。

```
tree (レベル, サイズ)
{
  if (最深レベルでないなら) {
    forward (幹または枝を描く);
    get (亀の状態を保存する);
    right (＋方向に角度をつける);
    tree (レベル－1, 少し小さなサイズ);
    set (亀を呼び戻す);
    right (－方向に角度をつける);
    tree (レベル－1, 少し小さなサイズ);
  }else (＝最深レベルなら)
    forward (緑色で葉を描く);
}
```

こうして作った木は，確かに木のようにも見えるが，どこか対称的すぎてつまらない。そこで，非対称に成長する木も考えてみた。tree2.cがそうである。幹の途中からも分岐するように仕組んだだけで，tree1.cと本質的にはまるっきり同じ。ところが，実行結果はかなり違うものとなった。個人的には，tree2.cで生成した木のほうが，自然ぽくて好きである。

実行方法はどちらでも同じで，起動すると再帰のレベルと分岐の角度を聞いてくる。いろいろと試してみてほしい。特にtree2はレベル5，角度20～30度あたりが個人的には好きである。

木の自動生成は，真面目に取り組めばなかなかに奥の深そうなテーマのようだ。今回は，分岐の際の枝のサイズの変化や，分岐の角度を一定にして木を生成したのだが，上手に乱数を使ってこれらの要素を撹乱してやれば，より自然物らしい表現が可能になると思う。それほど難しくない改造でできると思う。これは読者の演習問題として残しておくことにしよう。

ここで紹介した木のような図形は，フラクタル的な性質を持っているが，フラクタルとは呼ばれずにグラフタルと呼ばれているそうだ。グラフ理論方面からの言葉らしい。

さて，こうしたグラフィックデータの自動生成は，3次元コンピュータグラフィックにおいて，モデリングの強力な一手法となる。ちょいと角度の指定などを変えるだけで，単純計算の得意なコンピュータのこと，まったく違った木のデータがあっとい

対称形の木

非対称形の木

リスト4

```
==================  tree1.c  ==================
 1: /*木の自動生成(1)*/
 2:
 3: #include   <stdio.h>
 4: #include   <basic0.h>
 5: #include   "turtle.c"
 6:
 7: void   tree1( int, double, double );
 8:
 9: void   main()
10: {
11:    int   n;
12:    double   t;
13:    screen( 2,0,1,1 );
14:    printf( "再帰のレベルを教えてちょ：" );
15:    scanf( "%d", &n );          /*整数を入力させる*/
16:    pencolor( 1 );
17:    printf( "分岐の角度も教えてちょ：" );
18:    scanf( "%F", &t );          /*（倍精度）実数を入力させる*/
19:    set( 256.0, 512.0, 90.0 ); /*上向きに設定*/
20:    tree1( n, 128.0, t );       /*第一段の呼び出し*/
21:    return;
22: }
23:
24: void   tree1( n, length, t )
25: int    n;                /*再帰のレベル*/
26: double   length, t;   /*枝の長さ、角度*/
27: {
28:    double   x, y, arg;
29:
30:    if ( n>0 ) {                   /*最深レベルではないなら*/
31:       forward( length );        /*幹（枝）を伸ばし*/
32:       get( &x, &y, &arg );    /*現状を保存する*/
33:       right( t );                /*角度を変えて*/
34:       tree1( n-1, length/1.5, t );   /*レベルを1減らし、枝を短くして再帰呼び出し*/
35:       set( x, y, arg );         /*呼び出し前の状態を再現する*/
36:       right( -t );               /*反対方向に角度を変えて*/
37:       tree1( n-1, length/1.5, t );   /*再帰呼び出し*/
38:    } else {
39:       pencolor( 8 );            /*葉を描く*/
40:       forward( length );
41:       pencolor( 1 );
42:    }
43:    return;
44: }
```

図9　木の成長と変数の保護

う間にできてしまうからだ。そこでこの記事のしめくくりの余興として，木を3次元空間内に生成し，レイトレーシングでレンダリングするという無謀な試みをやってみた。

tree1の3D版がtree3，tree2の3D版がtree4である。テキストデータで出力する関係から，レンダラにはC-TRACEを使ったが，木の枝の数というのはあなどれないもので，すぐに数百～数千になってしまう。おかげで描画はあきれるほどのろい。これに挑戦したい方は心していただきたい。実行は簡単で，レイトレース用のデータファイルを出力するので，それをレンダラにかけるだけでよい。レベルは，最初2～3から始めたほうが無難であろう。5～6になるともはや遅いなんてものではなくなるから。アトリビュート周りはソース中にベタ書きなので，好きにいじってほしい。実行結果だけ見たいという方はカラーページか先月号の特集扉を引っ張り出して見ていただきたい。

tree3およびtree4の解説は割愛し，ヒント程度を図10に示す。このプログラムで使っている「自己参照する構造体」と再帰呼び出しの関係には，なかなかに深いものがあって面白く，けっこう高度なテクニックだと自負しているのだが，ある程度Cに慣れた人がソースとコメントを読んでもらえば十分というレベルのもので，ここでわかりやすく説明できる自信が僕にはない。K&Rを参照してほしい。

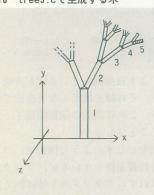
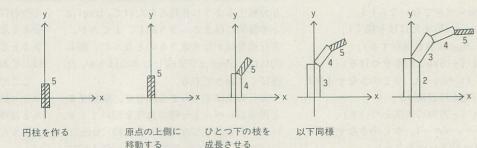
＊　＊　＊

再帰呼び出しというものが本質的に必要かどうかは議論の対象になろう。しかし，再帰はポイントを鋭く突いて使えば，これほどエレガントなものもない。K&Rのあるサンプルプログラムなどは，再帰を非常にうまく使った例である。あのレベルのプログラムが書けるようになるには，多少の修業が必要だろう。

本文中ちょくちょく登場した日本語Cのソース（？）は，プログラムもしくはアルゴリズムの，ほんのエッセンスを書いたものである。これと実際のソースリストの間には，多少のギャップがある。それは，変数の設定や，ポインタの調整や，座標の計算といった，細かい部分である。それは気合と根性と注意力だけが要求される世界である。それら汚れ仕事に埋もれてプログラムの本筋が見えなくなるといけないので，日本語Cを使わせてもらったのだ。プログラミングのあらすじを日本語Cで，コーディングの実際を本物のCで，というぐあいにして見比べると，もしかしたら勉強になるかもしれないと期待しつつ，今回はこのへんで失礼することにする。

リスト5

```
==================  tree2.c  ==================
  1: /*木の自動生成(2)*/
  2:
  3: #include   <stdio.h>
  4: #include   <basic0.h>
  5: #include   "turtle.c"
  6:
  7: void   tree2( int, double, double );
  8:
  9: void   main()
 10: {
 11:    int   n;
 12:    double   t;
 13:    screen( 2,0,1,1 );
 14:    printf( "再帰のレベルを教えてちょ：" );
 15:    scanf( "%d", &n );
 16:    pencolor( 1 );
 17:    printf( "分岐の角度も教えてちょ：" );
 18:    scanf( "%F", &t );
 19:    set( 256.0, 512.0, 90.0 );
 20:    tree2( n, 256.0, t );
 21:    return;
 22: }
 23:
 24: void   tree2( n, length, t )
 25: int   n;
 26: double   length, t;
 27: {
 28:    double   x, y, arg;
 29:
 30:    if ( n>0 ) {
 31:       forward( length/4.0 );           /*幹の長さの1/4のところから第一分岐*/
 32:       get( &x, &y, &arg );
 33:       right( -t );
 34:       tree2( n-1, length/2.0, t );  /*半分の大きさで再帰呼び出し*/
 35:
 36:       set( x, y, arg );
 37:       forward( length/4.0 );           /*1/2のところで第二分岐*/
 38:       get( &x, &y, &arg );
 39:       right( t );
 40:       tree2( n-1, length/2.0, -t );
 41:
 42:       set( x, y, arg );
 43:       forward( length/4.0 );           /*3/4の所で幹を伸ばす（これも再帰）*/
 44:       tree2( n-1, length/2.0, t );
 45:    } else {
 46:       pencolor( 8 );
 47:       forward( length );
 48:       pencolor( 1 );
 49:    }
 50:    return;
 51: }
```

図10　tree3.cで生成する木

枝5に注目する。
原点に作った円柱を左図の位置にまで移動するには：

```
struct MOVREV {
    char index;
    double rx, ry, my;
    struct MOVREV *next;
}
```

したがって，枝5の移動量は，枝1～4の成長を順に合計したものになる。
C-TRACEでは，枝4～枝1の順に記述するとよい。
tree3. cでは，枝の成長に伴う移動・回転の量をMOVREVという構造体に格納する。MOVREVは自己参照を行っていて，nextは根元に近い側に接続している枝へのポインタとなっている。たとえば枝5のnextは枝4を指す。ファイルを出力するときは，このnextを使って枝の先端から根元に向かって移動量をたどればよい。この構造体は再帰呼び出しとともにスタック上に生成され，次段呼び出しのときにそのポインタを引数として渡していく。

リスト6

```
==================  tree3.c  ==================
  1: /*3Dの木(レンダリングはC-TRACE)*/
  2:
  3: #include  <stdio.h>
  4:
  5: typedef struct MOVREV {
  6:    char      index;       /*幾つめの分岐かを示す通し番号*/
  7:    double   rx, ry, my; /*枝の移動量*/
  8:    struct   MOVREV   *next;   /*自己参照:構造体自身へのポインタ*/
  9: } MOVREV;
 10:
 11: void   tree3D( int, double, double, double, MOVREV* );
 12: void   mr_out( MOVREV* );           /*移動量をファイルに出力*/
 13: void   index_out( MOVREV* );    /*通し番号をファイルに出力*/
 14:
 15: FILE   *rdt;                 /*グローバル変数:レイトレースファイル*/
 16:
 17: void   main()
 18: {
 19:    int   n;
 20:    double   t;
 21:    MOVREV   mr;
 22:
 23:    printf( "ネストの深さを教えてちょ:" );
 24:    scanf( "%d", &n );
 25:    printf( "枝の角度も教えてちょ:" );
 26:    scanf( "%F", &t );
 27:    rdt=fopen( "TREE3.RDT", "wt" );
 28:
 29:    fprintf( rdt, "ratt¥n" );          /*アトリビュート部はベタ書き*/
 30:    fprintf( rdt, "branch_r¥n" );    /*枝の表面属性*/
 31:    fprintf( rdt, "¥t0.6 0.2 0.0¥n" );
 32:    fprintf( rdt, "¥t0.0 0.0 0.0¥n" );
 33:    fprintf( rdt, "¥t10.0 100.0¥n" );
 34:    fprintf( rdt, "leaf_r¥n" );        /*葉の表面属性*/
 35:    fprintf( rdt, "¥t0.0 0.8 0.0¥n" );
 36:    fprintf( rdt, "¥t0.0 0.0 0.0¥n" );
 37:    fprintf( rdt, "¥t10.0 100.0¥n" );
 38:    fprintf( rdt, "earth_r¥n" );       /*地面の表面属性*/
 39:    fprintf( rdt, "¥t0.6 0.6 0.2¥n" );
 40:    fprintf( rdt, "¥t0.0 0.0 0.0¥n" );
 41:    fprintf( rdt, "¥t10.0 100.0¥n" );
 42:    fprintf( rdt, "end¥n" );
 43:
 44:    fprintf( rdt, "tatt¥n" );
 45:    fprintf( rdt, "branch_t¥n" );    /*枝の透過属性*/
 46:    fprintf( rdt, "¥t0.0 0.0 0.0¥n" );
 47:    fprintf( rdt, "¥t1.0¥n" );
 48:    fprintf( rdt, "leaf_t¥n" )       /*葉の透過属性*/;
 49:    fprintf( rdt, "¥t0.0 0.0 0.0¥n" );
 50:    fprintf( rdt, "¥t1.0¥n" );
 51:    fprintf( rdt, "earth_t¥n" );      /*地面の透過属性*/
 52:    fprintf( rdt, "¥t0.0 0.0 0.0¥n" );
 53:    fprintf( rdt, "¥t1.0¥n" );
 54:    fprintf( rdt, "end¥n" );
 55:
 56:    fprintf( rdt, "shp¥n" );
 57:    fprintf( rdt, "earth¥n" );         /*地面は巨大な薄板*/
 58:    fprintf( rdt, "¥tbx¥t1000.0¥t1.0¥t1000.0¥n" );
 59:    fprintf( rdt, "¥t:my¥t-1.01¥n" );
 60:    fprintf( rdt, "¥t*r earth_r   *t earth_t¥n" );
 61:
 62:    mr.index=0;
 63:    mr.next=(MOVREV*)NULL;  /*ルートはポインタをNULLにすることで区別する*/
 64:    tree3D( n, 100.0, t, 3.0, &mr ); /*第一段の呼び出し*/
 65:    fprintf( rdt, "end¥n" );
 66:
 67:    fprintf( rdt, "lit¥n" );           /*光源その他もベタ書き*/
 68:    fprintf( rdt, "lt1¥n" );
 69:    fprintf( rdt, "¥tpa 1.0 -0.5 0.0¥n" );
 70:    fprintf( rdt, "¥t   10000.0¥n" );
 71:    fprintf( rdt, "¥t   1.0 1.0 1.0¥n" );
 72:    fprintf( rdt, "lt2¥n" );
 73:    fprintf( rdt, "¥tam 0.3 0.3 0.3¥n" );
 74:    fprintf( rdt, "end¥n" );
 75:
 76:    fprintf( rdt, "back¥t0.2 0.0 0.4¥n" );
 77:    fprintf( rdt, "allend¥n" );
 78:
 79:    fclose( rdt );
 80:    return;
 81: }
 82:
 83: void   tree3D( n, length, t, r, next )
 84: int   n;                          /*再帰レベル*/
 85: double   length, t, r; /*枝の長さ、分岐の傾き、枝の半径*/
 86: MOVREV   *next;              /*一段根元側にある枝の移動量*/
 87: {
 88:    MOVREV   mr;              /*再帰呼び出しと共に新しく作られる構造体*/
 89:
 90:    if ( n>0 ) {
 91:       fprintf( rdt, "branch" ); /*枝(幹)の生成*/
 92:       index_out( next );          /*通し番号付きで名前を出力*/
 93:       fprintf( rdt, "¥tcy¥t%f¥t%f¥t%f¥n", r, -length*0.5, r );/*枝(幹)を作
る*/
 94:       fprintf( rdt, "¥t:my¥t%f¥n", length*0.5 );
 95:       mr_out( next );               /*枝の移動量を出力*/
 96:       fprintf( rdt, "¥t*r branch_r  *t branch_t¥n" );
 97:
 98:       mr.index=1;       /*第一分岐*/
 99:       mr.rx=t;          /*幹に対して傾けて*/
100:       mr.ry=0.0;
101:       mr.my=length;     /*幹の先端から分岐する*/
102:       mr.next=next;
103:       tree3D( n-1, length*0.8, t, r*0.8, &mr );   /*再帰*/
104:
105:       mr.index=2;       /*第二分岐*/
106:       mr.rx=t;          /*幹に対して傾けて*/
107:       mr.ry=120.0;      /*120度回す*/
108:       mr.my=length;
109:       mr.next=next;
110:       tree3D( n-1, length*0.8, t, r*0.8, &mr );
111:
112:       mr.index=3;       /*第三分岐*/
113:       mr.rx=t;
114:       mr.ry=240.0;
115:       mr.my=length;
116:       mr.next=next;
117:       tree3D( n-1, length*0.8, t, r*0.8, &mr );
118:    } else {   /*コメントを取れば葉を出力するようになる*/
119: /*ここから   fprintf( rdt, "leaf" );
120:       index_out( next );
121:       fprintf( rdt, "¥tel¥t%f¥t%f¥t%f¥n", r*2.5, r*5.0, r*0.5 );
122:       fprintf( rdt, "¥t:my¥t%f¥n", r*5.0 );
123:       mr_out( next );
124:       fprintf( rdt, "¥t*r leaf_r       *t leaf_t¥n" );ここまで*/
125:    }
126:    return;
127: }
128:
129: void    mr_out( mr ) /*C-TRACEでは、移動量を*/
130: MOVREV  *mr;           /*先端から根元に向かって順に記述する*/
131: {
132:    while ( mr->next != (MOVREV*)NULL ) { /*ルートに達するまで繰り返す*/
133:       fprintf( rdt, "¥t:rxy¥t%f¥t%f¥n", mr->rx, mr->ry );
134:       fprintf( rdt, "¥t:my¥t%f¥n", mr->my );
135:       mr=mr->next;    /*一段根元に向かう*/
136:    }
137:    return;
138: }
139:
140: void   index_out( mr )   /*プリミティブ名を区別するための通し番号*/
141: MOVREV *mr;
142: {
143:    while ( mr->next != (MOVREV*)NULL ) {
144:       fprintf( rdt, "%d", (int)(mr->index) );
145:       mr=mr->next;
146:    }
147:    fprintf( rdt, "¥n" );
148:    return;
149: }
```

リスト7

```
void   tree3D( n, length, t, r, next )
int   n;
double   length, t, r;
MOVREV   *next;
{
   MOVREV   mr;

   if ( n>0 ) {
      fprintf( rdt, "branch" );   /*枝(幹)の生成*/
      index_out( next );
/**/      fprintf( rdt, "¥tcy¥t%f¥t%f¥t%f¥n", r, -length*0.4, r );
/**/      fprintf( rdt, "¥t:my¥t%f¥n", length*0.4 );
      mr_out( next );
      fprintf( rdt, "¥t*r branch_r  *t branch_t¥n" );

      mr.index=1;              /*第一分岐*/
      mr.rx=t;
      mr.ry=0.0;
/**/      mr.my=length*0.2; /*枝の長さの1/5のところから分岐する*/
      mr.next=next;
      tree3D( n-1, length*0.8, t, r*0.8, &mr );

      mr.index=2;              /*第二分岐*/
      mr.rx=t;
      mr.ry=120.0;
/**/      mr.my=length*0.4; /*2/5のところから分岐する*/
      mr.next=next;
      tree3D( n-1, length*0.8, t, r*0.8, &mr );

      mr.index=3;              /*第三分岐*/
      mr.rx=t;
      mr.ry=240.0;
/**/      mr.my=length*0.6; /*3/5のところから分岐する*/
      mr.next=next;
      tree3D( n-1, length*0.8, t, r*0.8, &mr );

/**/      mr.index=0;          /*第0分岐(枝を延長する)*/
/**/      mr.rx=0.0;
/**/      mr.ry=0.0;
/**/      mr.my=length*0.8; /*4/5のところから分岐する*/
/**/      mr.next=next;
/**/      tree3D( n-1, length*0.8, t, r*0.8, &mr );
   } else {
/*     fprintf( rdt, "leaf" );
      index_out( next );
      fprintf( rdt, "¥tel¥t%f¥t%f¥t%f¥n", r*2.5, r*5.0, r*0.5 );
      fprintf( rdt, "¥t:my¥t%f¥n", r*5.0 );
      mr_out( next );
      fprintf( rdt, "¥t*r leaf_r       *t leaf_t¥n" );*/
   }
   return;
}
```

特集2　Cプログラミング応用編

8ビットでもC言語を

αCで書く正規表現

Motohashi Jun
本橋　純

CP/M上で動作するC言語のなかではもっとも広まっていると思われるαC（BDS Cのサブセット）を使って文字列の正規表現をプログラムしてみましょう。サンプルはXCなどでも実行可能です。

C言語特集だと，どうしてもX68000が主流となって8ビットマシンは陰の存在となってしまいます。しかしご安心なされ。私が今回使うのはCP/M-80上で走るCのなかでもかなりメジャーなαC（BDS-C）です。これなら，MZ, X1ユーザーで使えますね（当然，持っていないと使えませんが）。なお，ここでのプログラムはX68000でも動きます。

さて，αCは標準的なCから一部の機能が削除されたいわゆるサブセット版です。そのため，ちょっと凝ったことをしようとすると動作しないことがあります。この制限のおかげで私は題材に困ってしまい，挙げ句の果てに正規表現などというとんでもない題材をプログラムすることになってしまいました（おかげで苦労するする）。

正規表現とは?

正規表現と聞いて皆さんなにを思い浮かべるでしょうか。言語を表すための手段のひとつと答えておくのが無難でしょう（コンパイラ関係の本だと，言語ではなく字句と書いてあります）。でも，たいていはCP/MやHuman, MS-DOSなどで使える（あと，BASICでも使えたか）ワイルドカードのもっとすごいやつとか，UNIX上でお目にかかれる摩訶不思議な表現という程度でしか知らないと思いますので，ここでは正規表現の定義を述べておくことにします。読んでいて頭が痛くなったら，読み飛ばしてもらってもかまいません。まずは定義の下準備として言葉の説明（定義）から。

・記号（文字）

a, b, c……や，あ, い, う……のように表されるものを記号と呼びます（文字とも呼ばれます）。

・アルファベット

記号の集合のことをアルファベットと呼びます。たとえば，記号a, bからなるアルファベットは{a, b}となります（{}は集合ということ）。日常でいわれているアルファベットとは{a, b, c, ……, z}などです。

・語（文字列）

あるアルファベットに属する記号を有限個並べたものを，そのアルファベットの上の語といいます。たとえば，アルファベットΣ＝{a, b}の上の語は，

a, b, baab, abaaba, ……

のようになります。それと，記号の個数が0のときの語は空語といい，εで表すことにします（計算機上でのヌルストリングを連想するとよいでしょうね）。

・言語

あるアルファベットΣ上の語の任意の集合をΣ上の言語といいます。例として，Σ＝{a, b}のときのΣ上の言語は，

{ε, a, b, baab, abaaba, ……}

となります。なお，空集合はφで表すことにします。

・連接

同アルファベット上の2つの語を接続して新たな語を作ることを，語の連接といいます。たとえば，

アルファベットΣ＝{あ, い}
語A＝あいあ, B＝いい

とすると，AとBの語の連接ABは，

AB＝あいあいい

となります。

また，2つのアルファベット上の言語を接続してできた新しい言語を，言語の連接といいます。言語の連接は2つの言語に含まれる任意の語を連接して作った語の集合となります。たとえば，

Σ1＝{い, ろ, は}, Σ2＝{に, ほ, へ}
言語L1＝{いろい, いはろ}
言語L2＝{ほへ, にへ}

としたとき，L1とL2の言語の連接L1L2は，

L1L2＝{いろいほへ, いろいにへ,
　　　いはろほへ, いはろにへ}

のようになります。

・閉包

言語Lを自分自身に任意回連接したものを言語の閉包といい，L*で表します。

L*＝{ε} ∪L∪LL∪LLL∪……

たとえば，L＝{mz, x}とすると，

L*＝{ε, mz, x, mzmz, mzx, mzmz
mz……}

となります。

さて，これらを踏まえたうえで正規表現の定義は次のようになります。

・正規表現

アルファベットΣ上の正規表現と正規表現が表す集合（言語）は，

1)　φは正規表現で空集合φを表す。
2)　εは正規表現で{ε}を表す。
3)　Σ上の各要素aについてaは正規表現であり，{a}を表す。
4)　s, tを正規表現とし，それらが示す集合をS, Tとすると，
　　(s+t) はS, Tの和集合S∪T
　　(st) はS, Tの連接ST
　　(s*) はSの閉包S*
　を表し，それらは正規表現である。
5)　1)～4)を有限回用いて導かれるもの以外は正規表現ではない。

わかりづらいと思うので例を挙げると，Σ＝{a, b}とすると，a, (a＋b), (ab)などは正規表現で，それぞれ集合{a}, {a, b}, {ab}を表しています。

また，Σ＝{X, 0, 6, 8}のとき

(X68)(0*)＝{x68}{ε, 0, 00, 000……}
　＝{x68, x680, x6800, x68000, …}

のようになります。

こうしてみると基本は和集合，連接，閉包ですが，次で述べるように現実で用いられている正規表現は便宜的な記号を用いて機能付加をしています。

Emacsの正規表現

ここでは実際に正規表現をプログラミングするわけですが，定義そのものをそのままプログラミングしても面白くないので，世の中で活躍しているものについて作ることにします。そこで今回は，UNIX上の2大エディタのひとつである（と枕詞をつけておこう）emacsで用いられている正規表現を作ってみます。UNIXのシェルのものでなくなぜemacsなのかというと，マニュアルを持っているから，という単にそれだけ

の理由からです。emacsでの正規表現の構文は結構多いので，ここではその一部分をプログラミングすることにします。

emacs上の正規表現で，‘^’，‘.’，‘*’，‘+’，‘?’，‘[’，‘]’，‘¥’の文字は特別な意味を持ちます。これ以外の文字は通常の文字です。ただし，‘¥’が前にあるときは少々違ってきます。

通常の文字は単純な正規表現で文字そのものにマッチします。たとえば，‘x’は文字列‘x’にマッチする正規表現です。また，‘m’，‘z’を連接させると‘mz’となりますので，文字列‘mz’にマッチするというようになるわけですね。これだけならS-OS上のエディタでも持っていますが，次から述べる特殊文字によって自由度は格段に広がります。

‘.’

これは改行文字以外のあらゆる1文字にマッチします。たとえば，‘M.T’は‘M’で始まり‘T’で終わる3文字にマッチします。つまり，‘MIT’や，“Monster-Attack-Team”の略称の‘MAT’，‘M#T’，……とマッチするわけです。

‘*’

これは定義の部分で述べた閉包の記号ですから，前の正規表現を可能な限り繰り返すことを意味します。当然，前の正規表現が1回もマッチしなくても，マッチするわけです。で，前の正規表現は最小のものが適用されます。たとえば，‘ab*’なら‘*’は‘b’について繰り返すというようになります。

さて，‘*’の動作は次のようになります。‘*’が現れるとまずマッチするものができるだけ長くなるようにします。次に残りの正規表現についてマッチさせるわけですが，これが失敗したときにはひとつ前の状態に戻り同様に残りがマッチするかを試みます。例を挙げて説明しましょう。

‘b*bc’を‘bbbc’にマッチさせることを試みます。まず，‘b*’によってすべてにマッチしたとすると残りは‘c’，正規表現側は‘bc’ですからマッチ失敗です。そこでひとつ戻ると‘bc’となりますからマッチします。つまり，ここでは‘b*’は‘bb’にマッチしたわけです。

‘+’

これは‘*’と同様の働きをしますが，この正規表現は少なくとも1回はマッチしなければならないところが‘*’とは異なります。たとえば，‘ab+c’は‘abc’，‘abbbc’などにはマッチしますが，‘ac’にはマッチしません。

‘?’

これは前の正規表現が0回か1回マッチしたときにマッチします。

‘^’

これは行の先頭の空文字にマッチします。それ以外はマッチしません。たとえば，‘^abc’は行頭にある‘abc’にマッチします。

‘$’

これは行の最後にマッチします。

‘¥’

これは特殊文字の機能をなくすために用いることと，次から述べる構成要素を作るのに用いられます。たとえば，‘¥$’は‘$’の文字にマッチします。うーん，バックスラッシュじゃないと気分が出ないなあ。¥だとエスケープ文字に見えないもんなあ。

‘¥n’

これはわかりますよね。改行文字にマッチします。

‘¥t’

これはタブ文字にマッチします。

‘¥|’

これは選択を表します。たとえば，‘cz¥|mz’なら‘cz’あるいは‘mz’にマッチします。これは定義でいうところの(s+t)の＋と同じですね。

‘¥(～¥)’

これはグループを作ります。この利用方法としては次のようなものがあります。なお，私のプログラムでは‘¥(¥(¥)¥)’のように入れ子は(問題点は多少あるものの)一応可能です。

1)　‘¥|’の選択範囲を定めます。つまり，‘Oh!¥(MZ¥|X¥)’は‘Oh!MZ’か‘Oh!X’にマッチします。

2)　‘*’などが利用できます。たとえば，‘a¥(ha¥)*’は‘a’，‘aha’，‘ahaha’……にマッチします。

3)　次で示す‘¥digit’に利用します。

‘¥digit’

‘¥(～¥)’によってマッチしたら，それ以降は‘¥(～¥)’は‘¥digit’で表現できます。つまり，“digit番目に出てきた‘¥(～¥)’でマッチしたテキスト”を意味します。‘digit’は‘1’から‘9’までで，‘¥(～¥)’が10個以上出てくる場合は最初の9個に対応します。例を挙げておくと，‘¥(test¥)¥1’は‘testtest’にマッチします。

‘[～]’

‘[’は文字集合の始め，‘]’は終わりを表します。たとえば，‘[xy]’は‘x’あるいは‘y’にマッチします。ということは‘¥(～¥|～¥)’のかたちで各要素が1文字の場合と同じ意味ですね。また，‘-’を用いることで文字範囲を書くことができます。たとえば，‘[a-z]’はすべての英小文字にマッチするということになります。

なお，前で述べてきた特殊文字はこの中では特殊ではなく通常の文字として扱われます。その代わり，別の特殊文字が存在します。それは‘]’，‘－’，‘^’です。これらの文字を文字集合に含みたい場合に‘]’，‘－’は最初に持っていき，‘^’は最初以外に持っていきます。また，‘－’は‘－－－’のように範囲のかたちでも書けます。ということは，すべてを含ませるときは‘[]^－－－]’のようにすればよいわけです。

‘[^～]’

‘[’の直後に‘^’がくるときは文字の補集合を意味します。つまり，内部に書いた文字以外にマッチするわけです。たとえば，‘[^a－zA－Z0－9]’は英数字以外にマッチします。

解説

私は100行以上のCのプログラムを作るのは2，3回目といったCビギナーなので，まだまだ修業が足りません。そのため，プログラムは結構見苦しいところだらけだと思いますが勘弁してください。Cでの文字列の扱い方を簡単に述べておいてから，プログラムの説明をすることにします。

CではBASICのように文字列型という型はありません。文字(char)型の配列で文字列を扱います。その際に文字データの後ろに‘¥0’がくっつきます。これは文字列はこれで終わりですよということを示すための印です。例を挙げておくと，“xyz”が文字配列a[]に入っているときは，

a[0]=‘x’，a[1]=‘y’，a[2]=‘z’，a[3]=‘¥0’

のようになっています。つまり，n文字(n≧0)の文字列を格納するには長さn+1の配列が必要となります。

このように，Cでは文字単位が基本でそれを集めて文字列を作っています。文字列単位が基本のBASICとは考え方が逆ですね。ですから，当然文字列処理の方法も逆になります。

・文字単位の処理

文字列のk文字目の文字を得るとき，Cではa[k－1]で得られます。これに対しBASICでは，mid$(A$，k，1)のように関数を用いないと得られません。

・文字列単位の処理

文字列の代入を行う際，Cでは関数strcpyを用いて，strcpy(a，“abc”)；のようにしなければ代入できません(初期設定以外のとき)。しかし，BASICでは代入文A$=“abc”ですみます。

さて，1文字の内容を得るのに，さっきはa［k−1］のように配列の添え字を用いています。現在位置が何文字目かを知らなければならないときはこれでもよいのですが，何文字目かそんなに重要でない場合はポインタを用いたほうがすっきり記述できます。

ポインタとはほかの変数のアドレスを持つ変数のことです。マシン語を知っている人はお馴染みだと思います。例を挙げて軽く説明しておきましょう。ここでは，文字ポインタをchar ＊pa;として，前述の“xyz”を入れたa［］を用いることにします。

まず，a［0］のアドレスをpaに代入します。これは，

pa ＝ &a［0］;

のようにします。‘&’は変数のアドレスを示す演算子ですね。なお，配列名は最初の要素の位置と同じなので上記の式はpa ＝ a;とも書けます。こうしてa［0］のアドレスを得たわけですが，内容を参照しないなら意味はありません。

そこで参照するためには，＊paのようにします。‘＊’をポインタの前につけると，そのポインタの内容を参照することができるわけです。ここではa［0］の内容‘x’が参照されるので，

c ＝ ＊pa;　(c はchar型とする)

ならば，cに‘x’が代入されます。

さて，次にa［1］を参照したいとします。参照方法のひとつは，＊(pa+1)ですが，これではa［0+1］と大差ありません。そこで，paの値を1加えて(つまり，pa++;をして) a［1］のアドレスにします。すると＊paは今度はa［1］の内容‘y’を参照しています。このように，いまのアドレスに加減算して，参照位置を変更できるので，ポインタは便利なわけです（もちろん，ほかにもポインタは役立つことだらけですが)。なお，a［k］のかたちは実際は＊(a+k)と同じことです。

さて，プログラムの説明です。まず，具体的な利用法を述べておきます。関数regexpは次のようになっています。

int regexp(char ＊＊x, char ＊＊y)

使用法は，呼び出す側で文字ポインタのポインタ（char ＊＊）を作っておいて，xの中に文字データのポインタを入れ，yの中に正規表現のポインタを入れてから呼び出す，あるいは文字ポインタに‘&’をつけてアドレスを渡すようにして呼び出すという2通りの方法があります。で，戻り値は正規表現の構文がおかしくてエラーになった場合2を，マッチに失敗した場合0を返します。そして，マッチした場合は1を返し，かつxの中にマッチ開始位置を，yの中にマッチ終了位置を入れて戻るようにしてあります。なお，この関数はあくまでテストのために作ったものなので，適当に変更して使うことをおすすめします。

関数regexpは単に処理本体側のために変数などの用意をするいわばお膳立て関数です。処理自身はregmainという関数が行います。この関数は，

int regmain (int i, int c,
char ＊＊pds, char ＊＊pde,
char ＊＊prs, char ＊＊pre)

と，引数が6つあります。以下で用途の説明をしましょう。

まず，int iについて。これには‘¥(～¥)’の出現回数を入れてあります。int cは‘?’，‘+’，‘＊’でマッチした回数が入っています。残りの4つの文字ポインタのポインタは呼び出すときにはデータの開始位置，終了位置，正規表現の開始位置，終了位置が入ります。そして，戻るときはマッチ開始位置，終了位置，正規表現の現在位置，終了位置が入れられ，関数の戻り値は上記のregexpと同じです。

さて，これらの引き数を一見すると文字ポインタのポインタ以外は用途がはっきりせず，いったいなんの役に立つんだと思われるかもしれません。しかし，iとcはともに関数regmainを再帰的に呼び出すときに意味を持つ引数なのです。この2つの引数がどう使われるかを，文字列データが“mz−2500”，正規表現が“¥（mz−¥）［0−9］＊”のときを例にしたものを（表1）に示しておきますので見てください。

これを見ればわかるとおり‘¥（～¥）’は内部を調べるのには再帰したほうが楽です（入れ子可能にしたため，ほとんどLISPのノリになってしまった)。

それに‘＊’にいたっては構文の説明のところで述べたように，マッチに失敗したらひとつ前に戻るといったことを行いますから，処理に再帰を利用したほうがプログラミングは格段に楽になります。しかし，再帰にするとメモリ効率，速度の面では不利になりますので，メモリが多く取れない場合（特に8ビットマシン）や，速度重視の場合は使わないことに越したことはありません（今回は楽をしたいため再帰を使った)。

次にマッチに失敗したときの処理を述べておきます。まず，マッチに失敗したら正規表現の残りをサーチして‘¥｜’があるかどうかを調べます。存在したら，データ位置は最初にマッチした次の位置へ（一度もマッチしてなかったら開始位置の次）に戻し正規表現は‘¥｜’の次からとなります。存在しなかったら次の処理を行います。

データの終了位置にきたならば，最後まで調べて失敗したのだから，“失敗”で戻ります。データの途中ならデータ位置を最初にマッチした位置の次に戻し，正規表現も最初に戻し，そして‘¥（～¥）’でマッチした個数も初期化します。失敗の流れをデータ“X1t”，正規表現“HC¥｜SMC”で（表2）に示しておきます。

αCの使い方

αCでのコンパイル方法を軽一く説明しておきます。まず，ソースリストをエディタを使って入力します。その際にファイルの拡張子は“.C”にしておきます。次に“SAMPLE.C”をコンパイルしたいとし

表1　MZ-2500のマッチング

		ポインタ位置以降の内容		マッチ位置		
count	digit	データ	正規表現	開始	終了	コメント
	0	mz-2531	¥(mz-¥)[0-9]*			
	->1	mz-2531	mz-¥)	m		
	1	z-2531	z-¥)	m		
	1	-2531	-¥)	m		
	<-1	2531	¥)	m	-	¥)なのでマッチで戻る
	0	2531	[0-9]*	m	-	¥(～¥)が最初なので開始も返した
=>1	0	531	[0-9]*	5		2がマッチしたので再帰に入った
=>2	0	31	[0-9]*	3		
=>3	0	1	[0-9]*	1		
=>4	0	¥0	[0-9]*	¥0		0個でのマッチだから再帰は停止
4	0	¥0	¥0	¥0	1	¥0の手前の1が終了位置となり戻る
<=3	0			1	1	終了位置を返し続ける
<=2	0			3	1	
<=1	0			5	1	
	0			m	1	これで処理終了

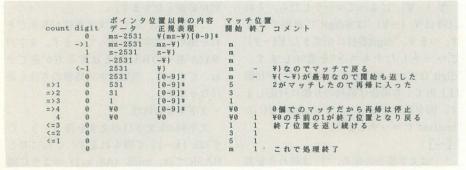



表2　マッチング失敗

ポインタ位置以降の内容		
データ	正規表現	コメント
X1t	HC¥|SMC	
X1t	SMC	マッチ失敗により¥|の次へきた
1t	HC¥|SMC	
1t	SMC	
t	HC¥|SMC	
t	SMC	
¥0	HC¥|SMC	
¥0	SMC	データ終了位置でマッチ失敗だから失敗で終了

ます。そしたら，

A>CC SAMPLE

とコマンド入力をします（ここでは拡張子は書かなくても平気です）。すると，

BDS Alpha－C Compiler v1.51 (partI)

35k elbowroom

BDS Alpha－C Compiler v1.51 (partII)

32k to spare

のようなメッセージが，エラーがない場合表示されます。エラーがあったらエラーメッセージが当然表示されます。で，無事コンパイル終了したらXCなどとは異なり実行形式のファイルはまだできあがっていません。リロケータブルバイナリファイルという長い名前のファイル“SAMPLE.CRL”ができているだけです。そこで，実行ファイルを得るために，

A>CLINK SAMPLE

というコマンド入力をするとエラーがなければ，

BDS Alpha－C Linker v1.51

Linkage complete

40K left over

のようなメッセージが表示されます。ここでは“.CRL”ファイルをリンカに通し，標準関数などとリンクをします。そうして，やっと“SAMPLE.COM”ができあがるわけです。めでたしめでたし。

＊　　＊　　＊

さて，これで皆さんも正規表現について多少なりとも理解が深められたでしょうか。読んだだけではわからないなら，入力してみていろいろと試してみることをおすすめします。その際に関数regmain中のprintf文をコメントではなくして途中の情報を表示させてみるとどんな処理をしているか，視覚的にもわかると思います。

ここで作ったプログラムは機能を追加したり（逆方向検索も簡単に実現できると思います），改造しまくるなり，美しく記述したり，あるいは虫を見つけたら退治するなりとどんどんいじくって活用してください。

参考文献

福村晃夫，稲垣康善著「オートマトン・形式言語理論と計算論」岩波書店

Richard Stallman著，竹内郁雄，天海良治監訳「GNU Emacsマニュアル」共立出版

リスト1　regexp.c

```
==================  regexp.c  ==================
  1: #include <stdio.h>       /* #include <bdscio.h> */
  2:
  3: #define EOB '¥0'
  4: #define EOS '¥0'
  5:
  6: #define UNMATCH 0
  7: #define MATCH   1      /* bit 0: MATCH/UNMATCH (1/0) */
  8: #define ERROR   2      /* bit 1: */
  9: #define FIRST   4      /* bit 2: FIRST/not (1/0) */
 10: #define MOV     8      /* bit 3: MOVE/STAY (1/0) */
 11:
 12: #define NOT     2      /* bit 1: 補集合/集合 (1/0) */
 13:
 14: /* 正規表現テスト */
 15:
 16: int regexp();
 17: int remain();
 18: char *getend();
 19:
 20: /* -_-_-_-_ main _-_-_-_- */
 21: main()
 22: {
 23:    char *wp;
 24:    char *x, *y;
 25:    char regdt[256], buffer[256];
 26:    int f;
 27:
 28:    while (1) {
 29:       printf("buffer?:¥n");
 30:       gets(buffer);
 31:       while (1) {
 32:          printf("regexp?:¥n");
 33:          gets(regdt);
 34:          if (regdt[0] == '¥0') break;
 35:
 36:          x = buffer;
 37:          y = regdt;
 38:          f = regexp(&x, &y);
 39:          printf("¥nbuffer: %s¥nregexp: %s¥n", buffer, regdt);
 40:          switch (f) {
 41:             case MATCH:
 42:                printf("MATCH!¥n");
 43:                for ( ; x != y + 1; putchar(*x++))
 44:                   ;
 45:                putchar('¥n');
 46:                break;
 47:             case UNMATCH:
 48:                printf("xxxxUNMATCHxxxx¥n");
 49:                break;
 50:             case ERROR:
 51:                printf("@@@@ERROR@@@@¥n");
 52:                break;
 53:          }
 54:       }
 55:    }
 56: }
 57:
 58: /* regular expression             */
 59: /*    MON Oct 23 1989  by motoha-j */
 60: /*<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>*/
 61: struct matchdata {
 62:    char *buf;          /* マッチ開始位置 */
 63:    char *bufend;       /* マッチ終了位置 */
 64: } w[10];
 65: int digmax;              /* digit最大値 */
 66:
 67: /* %%%%%%%%%%%%%%%%%%%%%% 正規表現テスト */
 68: int regexp(x, y)
 69: char **x;      /* in:評価文字列ポインタのポインタ out:マッチ開始位置 */
 70: char **y;      /* in:正規表現                     out:マッチ終了位置 */
 71: {
 72:    char *bufen, *regen;
 73:    int flag;
 74:
 75:    for (bufen = *x; *bufen != EOB; bufen++) /* 終了位置を求めている。*/
 76:       ;
 77:    for (regen = *y; *regen != EOS; regen++)
 78:       ;
 79:    digmax = 0;          /* digmax 初期化 */
 80:    flag = regmain(0, 0, x, &bufen, y, &regen);
 81:    *y = bufen;
 82:    return flag;
 83: }
 84:
 85: /* %%%%%%%%%%%%%%%%%%%% 処理本体 */
 86: int regmain(i, c, pds, pde, prs, pre)
 87: int i;                 /* digit */
 88: int c;                 /* '*','+','?'でのマッチ回数 */
 89: char **pds, **pde; /* 評価文字列ポインタポインタ 開始位置,終了位置 */
 90: char **prs, **pre; /* 正規表現ポインタポインタ 開始位置,終了位置 */
 91: {
 92:    char *d;           /* 評価文字列現在位置 */
 93:    char *r, *r0;      /* 正規表現現在位置、前回位置 */
 94:    int f;             /* flag */
 95:
 96:    d = *pds;
 97:    r = *prs;
 98:    f = FIRST | MOV;
 99:
100:    while (1) {
101:       {               /* 途中経過の表示（不要なら消して下さい。）*/
102:          int j;
103:          char *pt;
104:
105:          for (j = c; j--; )
106:             putchar(' ');
107:          printf("%d>[%d] %x: ", c, i, f);
108:          for (pt = d; pt != *pde; putchar(*pt++))
109:             ;
110:          putchar('¥t');
111:          for (pt = r; pt != *pre; putchar(*pt++))
112:             ;
113:          putchar('¥n');
114:       }                    /* 表示終了 */
115:
116:       r0 = r;
117:       f &= ~MATCH;
118:       /* #0.========================================   */
119:       if (r == *pre) {
120:          if (f & FIRST) return ERROR;
121:          *prs = r;
122:          *pde = d - 1;
123:          return MATCH;
124:       }
125:       /* #1.========================================   */
126:       switch (*r++) {
127:          /*--------------------------------------'.' */
128:          case '.':
129:             if ((*d) && *d != '¥n') f |= MATCH;
130:             break;
131:          /*--------------------------------------'^' */
132:          case '^':
133:             if (*(d-1) == '¥n' || (digmax == 0 && d == *pds)) {
134:                f |= MATCH;
135:                f &= ~MOV;
136:             }
137:             break;
138:          /*--------------------------------------'$' */
139:          case '$':
140:             if (*d != '¥n' && d != *pde) break;
141:             f |= MATCH;
142:             f &= ~MOV;
143:             break;
144:          /*-------------------------------------'[]' */
145:          case '[':
146:          {
147:             int f1;      /* local flag */
148:
149:             if (r == *pre) return ERROR;
```

```
150:          else if (*r == '^') {            /* 補集合か否か */
151:             r++; f1 = NOT;
152:          }
153:          else f1 = 0;
154:
155:          do {
156:             f1 &= ~MATCH;
157:             switch (*(r+1)) {
158:                case '-':
159:                   if (*r == '-' && *(r+2) == '-') { /* '---'の処理 */
160:                      if (*d == '-') f1 |= MATCH;
161:                      r += 3;
162:                      break;
163:                   }
164:                   else if (*(r+2) && *(r+2) != '-') { /*'α-β'の処理 */
165:                      if (*d >= *r && *d <= *(r+2))
166:                         f1 |= MATCH;
167:                      r += 3;
168:                      break;
169:                   }
170:                default:
171:                   if (*r++ == *d) f1 |= MATCH;
172:                   break;
173:             }
174:             if (f1 && MATCH)
175:                for (; *r != ']'; r++)
176:                   ;
177:          } while (r != *pre && *r != ']');
178:
179:          if (*r++ == *pre)
180:             return ERROR;
181:          if (f1 & NOT)            /* 補集合の時のフラグの反転 */
182:             f1 ^= NOT | MATCH;
183:          if (d != *pde)           /* メインのフラグに情報を渡す */
184:             f |= f1;
185:          break;
186:       }
187:       /*--------------------------------------'¥¥' */
188:       case '¥¥':
189:          switch (*r) {
190:             /*------------------------------'¥n' */
191:             case '¥n':
192:                if (*d == '¥n') f |= MATCH;
193:                break;
194:             /*------------------------------'¥t' */
195:             case '¥t':
196:                if (*d == '¥t') f |= MATCH;
197:                break;
198:             /*------------------------------'¥(' */
199:             case '(':
200:             {
201:                int ib;    /* 局所的digit */
202:                char *reb; /* '¥)' を終了位置とするための変数 */
203:
204:                if (digmax <= 9) digmax++;
205:                ib = digmax - 1; /* 引数の設定 */
206:                w[ib].buf = d;
207:                w[ib].bufend = *pde;
208:                reb = getend(++r);
209:                if (*reb == EOB) return ERROR;
210:
211:                switch (regmain(ib + 1, c, &w[ib].buf, &w[ib].bufend, &r,
&reb)) {
212:                   case MATCH:
213:                      d = w[ib].bufend; /* '¥(¥)'内部でのマッチ終了位置 */
214:                      if (f & FIRST) {
215:                         *pds = w[ib].buf;
216:                         f &= ~FIRST;
217:                      }
218:                      f |= MATCH;
219:                      break;
220:                   case UNMATCH:
221:                      r--;
222:                      break;
223:                   case ERROR:
224:                      return ERROR;
225:                   }
226:                break;
227:             }
228:             /*------------------------------'¥)' */
229:             case ')':
230:                if (i == 0) return ERROR;
231:             /*------------------------------'¥|' */
232:             case '|':
233:                if (f & FIRST) return ERROR;
234:                *prs = r;
235:                *pde = d - 1;
236:                return MATCH;
237:             /*---------------------------------- */
238:             default:
239:                /*-------------------------'¥digit' */
240:                if (*r >= '1' && *r <= '9') {
241:                   if (digmax < *r - '0') return ERROR;
242:                   {
243:                      char *dgs, *dge;
244:
245:                      dgs = w[*r - '1'].buf;
246:                      dge = w[*r - '1'].bufend;
247:                      f |= MATCH;
248:                      for (; dgs != dge + 1 && (f & MATCH); d++)
249:                         if (d == *pde || *d != *dgs++)
250:                            f &= ~MATCH;
251:                      d--;
252:                      break;
253:                   }
254:                }
255:                else if (*r == *d) f |= MATCH;
256:                break;
257:          }
258:          r++;                 /* end-of-'¥¥' */
259:          break;
260:       /*--------------------------------------other-char */
261:       default:
262:          if (*d == *(r-1)) f |= MATCH;
263:          break;
264:    }                                    /* end-of-switch */
265:
266:    /* #2.========================================   '*+?'-mode */
267:    switch (*r++) {
268:       /*---------------------------------------'+' */
269:       case '+':
270:          if (!(f & MATCH) && c == 0)
271:             break;
272:       /*---------------------------------------'?' */
273:       case '?':
274:          if (*(r-1) == '?' && c >= 1 && (f & MATCH)) {
275:             f &= ~MOV;
276:             break;
277:          }
278:       /*---------------------------------------'*' */
279:       case '*':
280:          if (f & MATCH)
281:          {
282:             char *dsb;
283:
284:             dsb = d + 1;
285:             switch (regmain(i, c + 1, &dsb, pde, &r0, pre)) {
286:                case MATCH:
287:                   if (f & FIRST) {
288:                      *pds = d; f &= ~FIRST;
289:                   }
290:                   return MATCH;
291:                case UNMATCH:
292:                   if (*(r-1) != '?') *pds = r;
293:                   break;
294:                case ERROR:
295:                   return ERROR;
296:             }
297:          }
298:          f |= MATCH;
299:          f &= ~MOV;
300:          break;
301:       /*---------------------------------------- */
302:       default:
303:          r--;
304:          break;
305:    }                                        /* end-of-'+*?' */
306:    /* #3.========================================   match&move */
307:    switch (f & MATCH) {
308:       case MATCH:
309:          if (f & FIRST) {
310:             *pds = d;
311:             f &= ~FIRST;
312:          }
313:          if (f & MOV) d++;
314:          else f |= MOV;
315:          break;
316:
317:       case UNMATCH:
318:          for   (; ;) {
319:             if (*r == '¥¥' && *(r+1) == '|') {
320:                r += 2;
321:                d = *pds;
322:                break;
323:             }
324:             else if (r++ == *pre) {
325:                if (d++ == *pde || i) {
326:                   *prs = *pre;
327:                   return UNMATCH;
328:                }
329:                if (!(f & FIRST)) {
330:                   f |= FIRST;
331:                   d = *pds; d++;
332:                }
333:                (*pds)++;
334:                r = *prs;
335:                digmax = i;
336:                break;
337:             }
338:          }
339:          break;
340:    }                 /* end-of-match&move */
341: }                    /* end-of-while(1) */
342: }
343:
344:
345: /* %%%%%%%%%%%%%%%%%%%% '¥(¥)'の対応位置を得る。*/
346: char *getend(r)
347: char *r;
348: {
349:    int nest;
350:
351:    nest = 1;
352:    while (nest) {
353:       switch (*r++) {
354:          case EOS:
355:             return --r;
356:          case '[':
357:             if (*r == EOS) break;
358:             for (r++; *r != ']' && (*r); r++)
359:                ;
360:             break;
361:          case '¥¥':
362:             switch (*r++) {
363:                case EOS: return --r;
364:                case '(': nest++; break;
365:                case ')': nest--; break;
366:             }
367:             break;
368:       }
369:    }
370:    return r;
371: }
```

ここまでできる

GCCに見る最適化

Nakamori Akira
中森　章

68000MPUではUNIX上のソフトウェアも簡単に走ります。GCCはかなり先端をいく最適化コンパイラです。浮動小数点などを除けば，XC比2～3倍の性能を示すことさえあります。ここでは最適化を初めとしたGCCの魅力を見ていきましょう。

はじめに

皆さんはGNU（グニュ）という言葉を聞いたことがありますか。GNUとは，

GNU's Not Unix!

（GNUはUNIXでない！）

の略語で，UNIXと互換性があり，かつ無料で提供されるソフトウェア環境のことです。当然UNIXに付属するもろもろのソフトウェアツールはGNUでも提供されています。

GNUのソフトウェアでいちばん有名なものはemacsエディタでしょう。現在，UNIX上でemacsといえば本家を差し置いてGNUのemacs（あるいはそれを日本語化したnemacs）のことを指すくらい普及していますからね。GNUのemacsの機能があまりにも高かったためにUNIX本来のemacsが駆逐されてしまったのです。

そして，それと同じような状況がCコンパイラについても起こっています。従来のUNIX上のCコンパイラはコードの最適化をあまり行っていません（RISCは別*1)）。しかし，GNUのCコンパイラ（GCC）はそれなりの最適化を行っています。その結果，従来のCコンパイラを使用するよりも2～3割高い性能を得ることができるのです。もともとCコンパイラはUNIXでのプログラム開発の中心になるものですから，GNUにおいても非常な関心を集め，たえずその改良が活発に行われてきています。

また，GCCは多くの人がその恩恵に預かれるようにといろいろなマシンに移植されています。最近ではあのTOWNSにも移植されたそうですが，わがX68000も例外ではありません。最初にX68000に移植されたのは（おそらく）GCCのバージョン1.31（電脳倶楽部で配布されたやつ）で，現在ではバージョン1.34や最新版の1.36も出回っているようです*2)。XCがコードの最適化をほとんどやってくれないことに頭を悩ましていた私が初めてGCCを手にしたときの感動は言葉ではいい表せません。今回このGCCを紹介する機会を編集部からいただきましたので，思いつくままGCCの素晴らしさ（特に最適化）について語っていきたいと思います。

*1)　バージョン1.36でGCCはRISCチップであるMIPS社のR2000用にも移植された。しかし，GCCでコンパイルしたプログラムはMIPS社本来のCコンパイラでコンパイルしたものよりかなり実行速度が遅かった。

*2)　GCCの移植は案外簡単で，ソースファイルの内の（多くの場合）4つをターゲットのCPUに依存する記述に書き直せばよい。ただし，32ビットCPUでないと結構大変な作業になる。

ANSI仕様に準拠

C言語といえば最近の話題はANSI仕様でしょう。少し前まではANSI準拠と称しながら，その実はvoidとenum宣言および構造体を関数の戻り値にできることをサポートしただけのCコンパイラばかりでした。しかし，現在のパソコン上（正確にはMS-DOS上）のCコンパイラはほぼ完全なANSI準拠になってきているようです。遅れているのはワークステーションやミニコン上のCコンパイラで，これらは依然としてANSIに似ている程度にとどまっています（それにはいろいろな事情があるのですが）。

その点GCCはワークステーション上ではいち早くANSI仕様を取り込んだCコンパイラの部類に属します。おそらく現在まともなANSI仕様のCコンパイラはGCCしかないといっていいかもしれません（int型が16ビットのCがまともといえるだろうか）。

もっともGCCはANSI仕様だけでは満足せず独自の拡張を行っています*3)が，コンパイル時のオプション指定で完全なANSI準拠のCコンパイラとしても旧仕様のCコンパイラとしても使用できるようになっています。ただ，この原稿はANSI準拠の紹介を主目的としていませんから，リスト1にANSIで規定された機能を用いたプログラム例を示すだけにしておきます。コメントを付けておきましたからどういう機能を試しているのかはわかるでしょう。このプログラムをコンパイルするためには－ansiオプションを忘れないでください（そうでな

GCCの入手方法

GNUはパブリックドメインソフトウェア群ではありません。当然，市販ソフトでもありません。GNUは広く配布されることを旨としたフリーソフトウェア群です。簡単な配布条件を満たせば，自由に移植・複製・配布・改良できます。

GNU配布の多くは手渡しまたは通信で行われています。満開製作所発行の電脳倶楽部1989年6月号のオマケとして配布されたこともあります。ですから，GCCをお持ちでない方で入手を希望される場合は知人をあたるか，通信によって入手します。

フリーソフトウェアという性格からか，現在X68000用として出回っているものだけでもたくさんのバージョンが存在します。元々のバージョンの違い，移植者の違い，バグフィックスなどによる移植バージョンの違いなどが考えられます。基本的な部分では大差はありませんが，細かな部分で違いがありますので注意してください。

また，GCCはそれぞれの機種のCコンパイラが持つライブラリに依存していますのでX68000で使用する場合はXCが必要になります。そのほか，GCCはもともとUNIX用の開発システムですから，通常のパソコン用Cコンパイラとは大きさが違います。コンパイラ本体で540Kバイト程度の大きさがありますので，メインメモリが1Mバイトのマシンでは控えめにいっても苦しいものがあります。

すべてのGNUウェアは配布時にGNU声明などのドキュメントが付属されているはずですから，ソースコードの入手法などについてはそれらを参照してください。

いとトライグラフが理解されない)。GCCのバージョン1.31と1.34で動作を確認ずみです。これでやっとK&Rの第2版に載っているプログラムを動かせますね*4)。

> *3) GCCではANSIにない仕様として64ビット整数型 (long long int型) をサポートしている。long long int型用の加減乗除などのライブラリが提供されるようになったのはバージョン1.36からである。それ以前のバージョンでもライブラリを自作することでlong long int型を使用できる。
> *4) いまだにK&Rの第1版が売れ続けているのはANSI仕様を満たすCコンパイラが少ないせいであろう。私自身は第1版は説明が不十分であまりいい出来とは思わないのだが。

優れた最適化

GCCの優れている点は出力するコードの最適化がよく行われているということです。コンパイラの最適化の理論は昔からエイホとウルマンの教科書[1)]で紹介されています。しかし，その昔ながらの理論ですら実際に適用しているCコンパイラはほとんどありませんでした (少なくとも適用しているとは思えなかった)。

現在パソコンの世界ではCコンパイラの性能競争が激しく，ひと通りの最適化は行われているようですが，ワークステーションの世界では進歩がありません。一方，GCCは昔から意欲的に最適化を行ってきており，この点ほかのCコンパイラに一歩先んじているといえるでしょう*5)。ここではGCCが実際に行っている最適化をXCと比較しながら紹介していきたいと思います*6)。使用したCコンパイラはGCCのバージョン1.34とXCのバージョン1.01です。

テスト1:定数のたたみ込み

これはコンパイル時に計算可能な式をあらかじめ計算してしまう最適化です。たとえば，

x = 2*3;

という式がある場合，実行時に常に2*3を計算してxに代入するのは時間の無駄です。それよりも式を計算して，

x = 6;

としておけばxには直接6を代入するだけでよくなります。このように直接定数式をプログラム中に記述することはまれかもしれません。しかし，プログラムではプリプロセッサの#defineで定義された定数値と別の数値との演算は非常によく現れるのでこの最適化は重要になってきます。これはもっとも基本的な最適化です (ほとんど常識) からやってないコンパイラはまずないでしょう。

また，直接に式で表されていなくても，あらかじめ値を計算することで実行速度を上げられる場合もあります。たとえば，aというint型配列の，

a [i+1]

という要素を参照する場合，アドレスは，

aのアドレス + (i+1) *4

によって計算されます。このとき，

(aのアドレス + 4) + i*4

リスト1 ANSI準拠

```
=================  ANSI.C  =================
 1: /*
 2:    ANSIに準拠した機能の例
 3: */
 4: /**********************
 5: **  プロトタイプ宣言  *
 6: ***********************/
 7: int   strcmp(char *s1, char *s2);  /* 有限個の引数 */
 8: void  printf(char *format, ...);   /* 不定個の引数 */
 9:
10: /************************
11: **  プリプロセッサ関連  *
12: *************************/
13: /* (1) #             マクロの引数に"つき文字列を与える */
14: /* (2) ##            マクロの引数を連結する            */
15: /* (3) __LINE__,...       定義ずみのマクロ                  */
16:
17: #define mess(that) "Can't you see that " #that "?¥n"
18: #define cat(x,y)   x ## y
19:
20: void sharps(void)
21: {
22:    printf(mess(You are sweet));        /* 文字列の引数 */
23:    printf("cat(1,2)=%d¥n", cat(1,2));/* 引数の連結   */
24: }
25:
26: void predefined(void)
27: {
28:    printf("__LINE__=%d¥n",__LINE__);/* 行番号       */
29:    printf("__FILE__=%s¥n",__FILE__);/* ファイル名   */
30:    printf("__DATE__=%s¥n",__DATE__);/* 日付         */
31:    printf("__TIME__=%s¥n",__TIME__);/* 時間         */
32:    printf("__STDC__=%d¥n",__STDC__);/* ANSI準拠か? */
33: }
34:
35: /*****************************************************
36: **  関数の宣言(プロトタイプ宣言も同時に行われる)  *
37: ******************************************************/
38: double add_inc(double x, double y)
39: {
40:    return(x+y+1);
41: }
42:
43: /**************************************************
44: **  トライグラフ(3文字で特殊な1文字を表現)  * こんなもの必要なの?
45: ***************************************************/
46: void trigraph(void)                  /* ??/ は ¥ なのでエスケープが必要 */
47: {
48:    printf("??= → # ; ¥??/ → ¥¥ ; ??' → ^ ¥n");
49:    printf("??( → [ ; ??) → ] ; ??! → | ¥n");
50:    printf("??< → { ; ??> → } ; ??- → ~ ¥n");
51: }
52:
53: /*******************
54: **  文字列の連結  *  プリプロセッサの # と同時に使うことが多い
55: ********************/
56: void catstr(void)
57: {
58:    char *s = "I think that "  /* 分けて書いた文字列が連結される */
59:              "that that "
60:              "that is used there "
61:              "is wrong!"  "¥n" ;
62:    printf(s);
63: }
64:
65: /******************
66: **  signed char  *
67: *******************/
68: signed char sc;
69:
70: /*************
71: ** 16進定数  *
72: **************/
73: char          a='¥x2f' ;    /* ¥x2f は10進数の47 */
74: unsigned char b='¥xAB' ;    /* ¥xAB は10進数の171 (符号なし)*/
75: signed   char c='¥xAB' ;    /* ¥xAB は10進数の-85 (符号つき)*/
76:
77: /*******************************************************
78: **  実行してみる ; -ansi オプションでコンパイルする  *
79: ********************************************************/
80: void main(void)
81: {
82:    sharps();        /* # と ## 演算子 */
83:    predefined();    /* 定義ずみマクロ */
84:    printf("add_inc(3,5)=%f¥n", add_inc(3,5));/* 引数はdoubleと宣言 */
85:    trigraph();      /* トライグラフ   */
86:    catstr();        /* 文字列の連結   */
87: }
```

と変形して（aのアドレス+4）をあらかじめ計算しておけばiに1を加えてから4倍するという操作が単純にiを4倍する操作に置き換わります。これはa[i+1]がループ内に現れるとき特に有効です。

リスト2に定数のたたみ込みを併うプログラムとコンパイル結果を示します。さすがにGCCもXCもちゃんと定数のたたみ込みを行っているようですね。ただXCには無駄なコードが多いのが気になりますが。

テスト2：無駄コードの削除

よく考えずにプログラムを書くと無駄なコードが発生してしまうことがあります。たとえば，

- return文よりも後ろの未到達コード
- 宣言しただけで使用しない変数
- 条件が常に偽のif文
- 実行されないループ

などです。これらのコードは最終的な実行結果に無関係ですから，それらに対するコードを生成することは実行速度の面からもプログラムサイズの面からも不利になります。このような無駄コードは削除されなくてはなりません。わざわざ無駄なコードを書く人はいないと思いますが，コンパイラのコード生成の仕方によっては，たまたま無駄なコードが生まれる場合もあります。また使用しない変数を宣言することは結構あるのではないでしょうか。

リスト3に無駄なコードを含むプログラムとそのコンパイル結果を示します。GCCではすべての無駄コードが削除されて非常にスッキリしたコードを生成していますが，XCはreturn文以降の未到達コードを削除する以外は馬鹿正直にコードを生成します。

テスト3：共通部分式削除

プログラム内では意図的にあるいは無意識に同じ式を何度か繰り返して書く場合があります。同じ値を持つ式が2カ所以上で現れる（これを共通部分式という）ならば，それを一度に計算したほうが演算回数を減らすことができます。これが共通部分式の削除と呼ばれる最適化です。これはいわゆる最適化コンパイラならば必ず行っているもっとも代表的な最適化です。たとえば，

x = a*b+c ;
y = a*b−c ;

という式があれば，a*bという共通部分式を認識して，

t = a*b ;
x = t−c ;
y = t+c ;

という式に変換するのが共通部分式の削除です。この変換で代入の回数は増えてしまいましたが演算回数（この場合は実行の遅い乗算の回数）を減らすことができました。リスト4に共通部分式の現れるプログラムとコンパイル結果を示します。GCCでは共通部分式がうまく削除されていますがXCではまったく考慮されていません。

ただ，このプログラムにはトリックがあります。a+b（加算）という式をa*b（乗

リスト2　定数のたたみ込み

```
==============  constfold.c  ==================
 1: /*
 2:    （最適化1）定数のたたみ込み
 3: */
 4: int     a[10];
 5: int     x;
 6: main()
 7: {
 8:         int i;
 9:         x=2*3;
10:         for(i=0;i<9;i++)
11:                 a[i+1]=x;
12: }
13: /*-------------------------------------
14:         gcc -O -S  でコンパイル
15: --------------------------------------
16: _main:
17:         link a6,#0
18:         moveq.l #6,d2              <= 2*3 を計算
19:         move.l d2,_x
20:         clr.l d1
21:         lea _a+4,a0                <= a+4 を計算
22: L5:
23:         move.l d1,d0
24:         asl.l #2,d0                <= 4*i
25:         move.l _x,(a0,d0.l)        <= (a+4)+4*i
26:         addq.l #1,d1
27:         moveq.l #8,d2
28:         cmp.l d1,d2
29:         bge L5
30:         unlk a6
31:         rts
32: --------------------------------------
33:         cc /O /S  でコンパイル
34: --------------------------------------
35: _main:
36:         BRA     L1
37: L20001:
38:         MOVE.L  -4(A6),D0        <= i
39:         ASL.L   #2,D0            <= 4*i
40:         ADD.L   #4+_a,D0         <= (a+4)+4*i
41:         MOVE.L  D0,A0
42:         MOVE.L  _x,(A0)          <= (a+4)+4*i
43: L6:
44:         ADDQ.L  #1,-4(A6)
45: L4:
46:         MOVE.L  -4(A6),D0
47:         CMP.L   #9,D0
48:         BLT     L20001
49: L3:
50:         UNLK    A6
51:         RTS
52: L1:
53:         LINK    A6,#-4
54:         MOVE.L  #6,_x            <= 2*3 を計算
55:         CLR.L   -4(A6)
56:         BRA     L4
57: ---------------------------------------*/
```

リスト3　無駄コード削除

```
==============  deadcode.c  ==================
 1: /*
 2:    （最適化2）無駄コード削除
 3: */
 4: int     x;
 5: main()
 6: {
 7:         int i,j;
 8:         j=0;                /* 未使用変数 */
 9:         i=0;
10:         if(0)   f(i);       /* 未到達コード */
11:         x=i;
12:         return;
13:         x+=10;              /* 未到達コード */
14: }
15: /*-------------------------------------
16:         gcc -O -S  でコンパイル
17: --------------------------------------
18: _main:
19:         link a6,#0
20:         clr.l _x                   <= x=0(=i)
21:         unlk a6
22:         rts
23: --------------------------------------
24:         cc /O /S  でコンパイル
25: --------------------------------------
26: _main:
27:         BRA     L1
28: L2:
29:         CLR.L   -8(A6)           <= j=0
30:         CLR.L   -4(A6)           <= i=0
31:         CLR.L   D0
32:         BEQ     L4
33:         MOVE.L  -4(A6),-(SP)
34:         JSR     _f               <= f(i)
35:         ADDQ.L  #4,SP
36: L4:
37:         MOVE.L  -4(A6),_x        <= x=i
38:         UNLK    A6
39:         RTS
40: L1:
41:         LINK    A6,#-8
42:         BRA     L2
43: ---------------------------------------*/
```

算）に置き換えるとGCCでも共通部分式の削除ができなくなってしまいます。これは乗算が関数によって実行される（MC68000に32ビット乗算命令はない）ために，一般的な最適化のアルゴリズムが使えなくなるためと考えられます（関数はいろいろな副作用を伴うのでそれを含む式の最適化は難しい）。乗算がある場合，試しに－m68020というオプションをつけてコンパイルしてみましょう。これはMC68020（32ビット乗算命令がある）用のコード（もちろんX68000上では実行できないが，コード生成は可能）を出力するためのオプションです。この場合は乗算を含む共通部分式もちゃんと削除されるようです（早くX68020かX68030が発売されないかなあ）。なお，残念ながら（？）現在X68000用に出回っているバージョン1.36の移植版では68020のコードが出せないものもあるようです。

テスト4：ループ内不変式の移動

ループ内にあって，値がループの繰り返しによっても変化しない式をループ内不変式といいます。このような式をループの繰り返しごとに計算し直すのは時間の無駄です。そこでループ内不変式はループに入る前に計算してしまうという考え方があります。これがループ内不変式の移動と呼ばれる最適化です。たとえば，

```
for (i=0；i<10；i++) {
    x=x+a；
    a=b−c+d；
}
```

というループがあるとき，(b−c+d) の値はループ内で不変です。このような場合は，

```
t=b−c+d；
for (i=0；i<10；i++) {
    x=x+a；
    a=t；
}
```

としてコンパイルすればループ内での計算回数を減らすことができます。リスト5にループ内不変式を持つプログラムとそのコンパイル結果を示します。GCCは期待どおりの最適化をやってくれていますが，XCではループ内不変式に関してはなんの最適化もやっていません。なお，リスト2のGCCのコンパイル結果で，ループ外で(aのアドレス+4) をa0レジスタに入れていたのもこのループ内不変式の移動です。

リスト4　共通部分式の削除

```
===============  cseelim.c  ==================
 1: /*
 2:    （最適化３）共通部分式の削除
 3: */
 4: int       x,y,z;
 5: int       a,b,c;
 6: main()
 7: {
 8:           x=a+b-c;
 9:           y=a+b+c;
10:           z=a+b;
11: }
12: /*-------------------------------------
13:           gcc -O -S  でコンパイル
14: --------------------------------------
15: _main:
16:           link a6,#0
17:           move.l _a,d0
18:           add.l _b,d0      <= d0: a+b
19:           move.l d0,d1
20:           sub.l _c,d1      <= (a+b)-c
21:           move.l d1,_x
22:           move.l d0,d1
23:           add.l _c,d1      <= (a+b)+c
24:           move.l d1,_y
25:           move.l d0,_z     <= (a+b)
26:           unlk a6
27:           rts
28: --------------------------------------
29:           cc /O /S  でコンパイル
30: --------------------------------------
31: _main:
32:           BRA      L1
33: L2:
34:           MOVE.L   _a,D0
35:           ADD.L    _b,D0     <= d0: a+b
36:           MOVE.L   _c,D1     <= d1: c
37:           SUB.L    D1,D0     <= (a+b)-c
38:           MOVE.L   D0,_x
39:           MOVE.L   _a,D0
40:           ADD.L    _b,D0     <= d0: a+b
41:           ADD.L    D1,D0     <= (a+b)+c
42:           MOVE.L   D0,_y
43:           MOVE.L   _a,D0
44:           ADD.L    _b,D0     <= d0: a+b
45:           MOVE.L   D0,_z
46: L3:
47:           UNLK     A6
48:           RTS
49: L1:
50:           LINK     A6,#0
51:           BRA      L2
52: --------------------------------------*/
```

リスト5　ループ内不変式の移動

```
===============  loopinv.c  ===================
 1: /*
 2:    （最適化４）ループ内不変式の移動
 3: */
 4: int       x;
 5: int       a,b,c,d;
 6: main()
 7: {
 8:           int i;
 9:           a=1;
10:           for(i=0;i<10;i++){
11:              x=x+a;
12:              a=b-c+d;       /* 不変式 */
13:           }
14: }
15: /*-------------------------------------
16:           gcc -O -S  でコンパイル
17: --------------------------------------
18: _main:
19:           link a6,#0
20:           moveq.l #1,d2
21:           move.l d2,_a
22:           clr.l d1
23:           move.l _b,d0
24:           sub.l _c,d0
25:           add.l _d,d0      <= d0: b-c+d
26: L5:                        ---- ここからループ
27:           move.l _a,d2
28:           add.l d2,_x      <= x=x+a
29:           move.l d0,_a     <= a=(b-c+d);
30:           addq.l #1,d1     ---- ここまでループ
31:           moveq.l #9,d2
32:           cmp.l d1,d2
33:           bge L5
34:           unlk a6
35:           rts
36: --------------------------------------
37:           cc /O /S  でコンパイル
38: --------------------------------------
39: _main:
40:           BRA      L1
41: L20001:                        ---- ここからループ
42:           MOVE.L   _x,D0
43:           ADD.L    _a,D0     <= d0: x+a
44:           MOVE.L   D0,_x     <= x=(x+a)
45:           MOVE.L   _b,D0
46:           MOVE.L   _c,D1
47:           SUB.L    D1,D0     <= d0: b-c
48:           ADD.L    _d,D0     <= d0: (b-c)+d
49:           MOVE.L   D0,_a     <= a=(b-c+d)
50: L6:                            ---- ここまでループ
51:           ADDQ.L   #1,-4(A6)
52: L4:
53:           MOVE.L   -4(A6),D0
54:           CMP.L    #10,D0
55:           BLT      L20001
56: L3:
57:           UNLK     A6
58:           RTS
59: L1:
60:           LINK     A6,#-4
61:           MOVE.L   #1,_a
62:           CLR.L    -4(A6)
63:           BRA      L4
64: --------------------------------------*/
```

テスト5：自動変数のレジスタ割り付け

C言語では使用頻度の高い変数を高速に参照するために（できるだけ）レジスタに割り付けるためのregister宣言がありました。しかし，そのようなことは人間がいちいち指示しなくてもコンパイラが自主的に行うのが正しいあり方ではないでしょうか。GCCでは自動変数をできるだけレジスタに割り付けるようなコード生成を行います。そのため，自動変数を使用する式を高速に計算することができます。静的変数は通常はレジスタに割り付けられませんがコンパイル時のオプション（-fforce-mem）でレジスタに割り付けることも可能です*7)。リスト6にレジスタ割り付けを行う例を示しましょう。当然ながら，XCではregister宣言がないので自動変数はメモリに割り付けられています。

テスト6：連続関数呼び出し時の引数ポップ

通常のCコンパイラで，

```
f (a,b) ;
```

という関数呼び出しをコンパイルすると次のようなコードが生成されます。

```
push b      ；引数をスタックに積む
push a      ；引数をスタックに積む
jsr  f      ；関数を呼ぶ
add  #8,sp  ；スタックを補正する
```

つまり，引数のプッシュによってずらしたスタックポインタを関数から戻ってきたあとで補正します。ところで，もし関数の呼び出しが，

```
f (a,b);
g (x,y);
```

のように連続する場合は，

```
push b      ；引数をスタックに積む
push a      ；引数をスタックに積む
jsr  f      ；関数を呼ぶ
add  #8,sp  ；スタックを補正する
push y      ；引数をスタックに積む
push x      ；引数をスタックに積む
jsr  g      ；関数を呼ぶ
add  #8,sp  ；スタックを補正する
```

となりますが，この命令列をよく眺めると，

```
jsr f
```

と，

```
push y
```

のあいだにある，

```
add #8,sp
```

はあまり意味のないコードであるとわかります。このコードを省略すると関数gを呼ぶ前にスタックポインタの値が8だけずれたままになります。しかし，gから戻ってきたあとに，

```
adda.l #16,sp
```

によって一括してスタックポインタの補正をしてやればなにも不都合は起きません。

GCCはこういう方法で関数呼び出し後のスタックポインタの補正を省略して実行速度を上げています。これを示すプログラムがリスト7です。XCではそんなややこしいことはなにも考えられていませんね。

なお，リスト7のGCCのコンパイル結果では，unlk命令でスタックポインタが関数が呼び出された直後の値に復帰することを利用して，最終的なスタックポインタの補正まで省略しています（素晴らしい）。もっとも常にこんな最適化を行うと不都合が生じる場合があるかもしれないので，GCCではこのようなスタックポインタの一括しての補正を禁止するコンパイル時のオプション（－fno－defer－pop）も用意されています。これで万全ですね。

リスト6　自動変数のレジスタ割り付け

```
==============  regalloc.c  ==================
 1: /*
 2:    （最適化5）自動変数のレジスタ割り付け
 3: */
 4: main()
 5: {
 6:          int a,b,c,d,e;
 7:          a=b;
 8:          if(b==e){
 9:             c=a+e;
10:             d=b;
11:          }
12:          else {
13:             c=a-e;
14:             d=e;
15:          }
16:          b=c+e;
17:          e=b+d;
18:          f(a+b+c+d+e);
19: }
20: /*-----------------------------------
21:          gcc -O -S  でコンパイル
22: -----------------------------------
23: d3 の退避・回復以外にメモリ参照なし
24: _main:
25:          link a6,#0
26:          move.l d3,-(sp) <= d3 退避
27:          move.l d0,d3    <= d0:b ; d3:a
28:          cmp.l d0,a1     <= a1:e
29:          bne L2
30:          move.l d0,d2
31:          add.l a1,d2     <= d2:c=b+e=a+e
32:          move.l d0,d1    <= d1:d
33:          bra L3
34: L2:
35:          move.l d3,d2
36:          sub.l a1,d2     <= d2:c=a-e
37:          move.l a1,d1    <= d1:d
38: L3:
39:          move.l d2,d0
40:          add.l a1,d0     <= d0:b=c+e
41:          move.l d0,a1
42:          add.l d1,a1     <= a1:e=b+d
43:          move.l d3,a0    <= a0:a
44:          add.l d0,a0     <= a0:(a)+b
45:          add.l d2,a0     <= a0:(a+b)+c
46:          add.l d1,a0     <= a0:(a+b+c)+d
47:          pea (a1,a0.l)   <= push a1+a0=e+(a+b+c+d)
48:          jsr _f
49:          move.l -4(a6),d3<= d3 回復
50:          unlk a6
51:          rts
52: -----------------------------------
53:          cc /O /S  でコンパイル
54: -----------------------------------
55: _main:
56:          BRA      L1
57: L4:
58:          MOVE.L  -4(A6),D0
59:          MOVE.L  -20(A6),D1
60:          SUB.L   D1,D0
61:          MOVE.L  D0,-12(A6)
62:          MOVE.L  -20(A6),-16(A6)
63: L5:
64:          MOVE.L  -12(A6),D0
65:          ADD.L   -20(A6),D0
66:          MOVE.L  D0,-8(A6)        <= b=c+e
67:          MOVE.L  -8(A6),D0
68:          ADD.L   -16(A6),D0
69:          MOVE.L  D0,-20(A6)       <= e=b+d
70:          MOVE.L  -4(A6),D0        <= a
71:          ADD.L   -8(A6),D0        <= (a)+b
72:          ADD.L   -12(A6),D0       <= (a+b)+c
73:          ADD.L   -16(A6),D0       <= (a+b+c)+d
74:          ADD.L   -20(A6),D0       <= (a+b+c+d)+e
75:          MOVE.L  D0,-(SP)         <= push (a+b+c+d+e)
76:          JSR     _f
77:          ADDQ.L  #4,SP
78: L3:
79:          UNLK    A6
80:          RTS
81: L1:
82:          LINK    A6,#-20
83:          MOVE.L  -8(A6),-4(A6)    <= -8(a6):b ; -4(a6):a
84:          MOVE.L  -8(A6),D0
85:          CMP.L   -20(A6),D0       <= -20(a6):e
86:          BNE     L4
87:          MOVE.L  -4(A6),D0
88:          ADD.L   -20(A6),D0
89:          MOVE.L  D0,-12(A6)       <= -12(a6):c
90:          MOVE.L  -8(A6),-16(A6)   <= -16(a6):d
91:          BRA     L5
92: ----------------------------------*/
```

テスト7:ループの最適化

ループ内で配列要素を参照するときそれがループごとに連続した領域になることがしばしばあります。たとえば,

　　for (i=0;i <100;i++)

　　　a [i]=0;

というint型の配列を初期化するループを考えてみましょう。この場合，ループ内にあるa [i] という配列要素のアドレスは,

　　aのアドレス + i*4

によって計算されます。つまり1回の乗算と1回の加算が必要です。しかし，ポインタ変数を用いた,

　　p=&a [0] ;

　　for (i=0;i <100;i++) {

　　　*p = 0;

　　　p++;

　　}

というループで同じことを行う場合は配列要素のアドレスをわざわざ計算する必要はありません。p++によって次の要素のアドレスを計算してはいますが，これは加算1回で行うことができます(MC68000のポストインクリメント・アドレッシングを用いればその必要もない)。これはループ内の計算の「強さ」を減少したことになります(当然実行速度は上がる)。これは演算強度の軽減(ストレングスリデュース)と呼ばれる最適化です。GCCは通常はこのような最適化は行いませんがコンパイル時に,

　　-fstrength-reduce

というオプションをつけることで演算強度の軽減を行うことができます。リスト8に演算強度の軽減を利用したループの最適化を行うプログラム例を示します。ここではもはやXCとの比較をやめて-fstrength-reduceオプションがある場合とない場合のGCCで比較を行っています。リスト8ではループの1回の繰り返しごとに配列要素を示すポインタの値が4ずつ増えていくので少し複雑なコードになっています。

テスト8:フレームポインタの削除

関数の入り口では通常のCコンパイラではスタックフレームを生成し(MC68000ではlink命令を使う)，フレームポインタからの相対位置を指定して引数や局所変数を参照します。たとえば,

リスト7　引数ポップ

```
============== parampop.c ==================
  1: /*
  2:     (最適化6) 連続関数呼び出し時の引数ポップ
  3: */
  4: main()
  5: {
  6:         int a,b,c;
  7:         a=1;
  8:         b=a+1;
  9:         c=b+1;
 10:         f(a,b);
 11:         g(b,c);
 12:         h(a,b,c);
 13: }
 14: /*-------------------------------------
 15:         gcc -O -S  でコンパイル
 16: -------------------------------------
 17: 関数呼び出し後にスタックポインタの補正をしない
 18: _main:
 19:         link a6,#0
 20:         movem.l d3/d4/d5,-(sp)
 21:         moveq.l #1,d3    <= a
 22:         moveq.l #2,d4    <= b 定数のたたみ込み
 23:         moveq.l #3,d5    <= c 定数のたたみ込み
 24:         move.l d4,-(sp) <= push b
 25:         move.l d3,-(sp) <= push a
 26:         jsr _f
 27:         move.l d5,-(sp) <= push c
 28:         move.l d4,-(sp) <= push b
 29:         jsr _g
 30:         move.l d5,-(sp) <= push c
 31:         move.l d4,-(sp) <= push b
 32:         move.l d3,-(sp) <= push a
 33:         jsr _h
 34:         movem.l -12(a6),d3/d4/d5
 35:         unlk a6
 36:         rts
 37: -------------------------------------
 38:         cc /O /S  でコンパイル
 39: -------------------------------------
 40: _main:
 41:         BRA     L1
 42: L2:
 43:         MOVE.L  #1,-4(A6)           <= -4(a6):a
 44:         MOVE.L  -4(A6),D0
 45:         ADDQ.L  #1,D0               <= a+1
 46:         MOVE.L  D0,-8(A6)           <= -8(a6):b
 47:         MOVE.L  -8(A6),D0
 48:         ADDQ.L  #1,D0               <= b+1
 49:         MOVE.L  D0,-12(A6)          <= -12(a6):c
 50:         MOVE.L  -8(A6),-(SP)        <= push b
 51:         MOVE.L  -4(A6),-(SP)        <= push a
 52:         JSR     _f
 53:         ADDQ.L  #8,SP               <= sp の補正
 54:         MOVE.L  -12(A6),-(SP)       <= push c
 55:         MOVE.L  -8(A6),-(SP)        <= push b
 56:         JSR     _g
 57:         ADDQ.L  #8,SP               <= sp の補正
 58:         MOVE.L  -12(A6),-(SP)       <= push c
 59:         MOVE.L  -8(A6),-(SP)        <= push b
 60:         MOVE.L  -4(A6),-(SP)        <= push a
 61:         JSR     _h
 62:         LEA     12(SP),SP           <= sp の補正
 63: L3:
 64:         UNLK    A6
 65:         RTS
 66: L1:
 67:         LINK    A6,#-12
 68:         BRA     L2
 69:         .DATA
 70:         .EVEN
 71:         .END
 72: -------------------------------------*/
```

リスト8　ループの最適化

```
============== strength.c ==================
  1: /*
  2:     (最適化7) ループの最適化
  3: */
  4: int     x;
  5: int     a[10];
  6: main()
  7: {
  8:         int i;
  9:         for(i=0;i<10;i++)
 10:             a[i]=4*i-x;
 11: }
 12: /*-------------------------------------
 13:         gcc -O -S  でコンパイル
 14: -------------------------------------
 15: _main:
 16:         link a6,#0
 17:         clr.l d1                      <= d1: i
 18:         lea _a,a0
 19: L5:
 20:         move.l d1,d0
 21:         asl.l #2,d0                   <= 4*i
 22:         move.l d0,d2
 23:         sub.l _x,d2                   <= (4*i)-x
 24:         move.l d2,(a0,d0.l)           <= a[i]=(4*i)-x
 25:         addq.l #1,d1                  <= i の差分
 26:         moveq.l #9,d2
 27:         cmp.l d1,d2
 28:         bge L5
 29:         unlk a6
 30:         rts
 31: -------------------------------------
 32:         gcc -O -S -fstrength-reduce  でコンパイル
 33: -------------------------------------
 34: _main:
 35:         link a6,#0
 36:         clr.l d1
 37:         lea _a,a0                     <= p= &a[0]
 38:         clr.l d0
 39: L5:
 40:         move.l d0,d2                  <= d0: 4*i
 41:         sub.l _x,d2                   <= (4*i)-x
 42:         move.l d2,(a0)+               <= *p=(4*i)-x ; p++;
 43:         addq.l #4,d0                  <= 4*i の差分
 44:         addq.l #1,d1                  <= i   の差分
 45:         moveq.l #9,d2
 46:         cmp.l d1,d2
 47:         bge L5
 48:         unlk a6
 49:         rts
 50: -------------------------------------*/
```

```
f (x, y)
int x, y;
 {
    return (x+y) ;
}
```

という関数はMC68000用のCコンパイラでは,

```
f:
    link a6,#0 ;フレームポインタをa6に
    move.l 8 (a6),d0 ;xをd0に
    add.l 12 (a6),d0 ;yをd0に加算
    unlk a6 ;スタックフレーム削除
    rts
```

とコンパイルされるでしょう。しかし，関数に局所変数がまったくない場合，link命令でフレームポインタを作らなくてもスタックポインタからの相対位置で引数を簡単に参照することができます。たとえば，上の場合，

```
f:
    move.l 4 (sp),d0 ;xをd0に
    add.l  8 (sp),d0 ;yをd0に加算
    rts
```

で十分用が足りてしまい，linkとunlkの2命令が節約できたことになります（時間にして約66クロック減少）。こんなうまい話を性能重視のGCCが見逃すはずがありません。GCCではコンパイル時に,

−fomit−frame−pointer

というオプションを指定することでスタックフレームの作成（linkとunlkの実行）をやめることができます。これを示すプログラム例がリスト9です。ここでもXCとの比較はやっていません。見てわかるように，関数内で局所変数を必要とする場合は必ずスタックフレームが作られるようです。

テスト9：関数のインライン展開

Cコンパイラの最適化のなかで究極の最適化のひとつは関数のインライン展開です。これは関数の本体をそれを呼び出す位置に埋め込んでしまい，関数呼び出しの時間を節約する最適化です。確かに実行速度は向上する（約40クロック節約）のですが，コードサイズは明らかに増加しますし，あえて関数として宣言されたものをコード内に埋め込んでもいいものかという疑問も残ります。また，分割コンパイルの前には無力になってしまいますし，関数のインライン展開はベンチマーク以外にどの程度有用であるかどうかははっきりしません。しかし，関数のインライン展開はコンパイラ技術としては興味深いものがあります。GCCでも簡単な関数については,

−finline−functions

というオプションをつけてコンパイルすることでインライン展開ができるようになっています（簡単な関数という定義はよくわかりませんが)。

リスト10に関数のインライン展開を行うプログラム例を示します。本当に関数が見事に埋め込まれていますね。リスト10ではインライン展開したあとには定数のたたみ込みの最適化がさらに行われています（引数が定数だと関数の戻り値が計算できてしまう)。

＊5） ワークステーション上のCコンパイラがANSI仕様にバージョンアップされないのは互換性の問題があるからである。Cコンパイラはあまりにも基本的なソフトなので簡単に置き換えてしまうことはできない。その点，OSにそれほど密着していないパソコン上のCコンパイラやワークステーション上のクロスコンパイラ（GCCはこの性格が強い）は最新の仕様を持たせたバージョンアップをしやすい。

＊6） XCはX68000の持っている機能をすべて実行できるように多数のライブラリを提供している点が大いに評価できる。しかしながら，コードの最適化についてはお世辞にも優れているとはいえない（高機能・低性能の典型例)。GCCのコードと比較することで，やればここまでできるという可能性を示して次のバージョンに反映されることを期待したい（現在開発中という噂もある)。

＊7） RISC用のCコンパイラが非常に性能がいいのは(ひと通りの最適化のほかに)大域変数のレジスタ割り付けがよく行われているからだと思ってよい。これはまじめ

リスト9 フレームポインタの削除

```
==============  frame.c  ===================
 1: /*
 2:    (最適化8)フレームポインタの削除
 3: */
 4: f(x,y)
 5: int x,y;
 6: {
 7:         return(x+y+10);
 8: }
 9:
10: g(x)
11: int x;
12: {
13:         int a[3];
14:         a[1]=f(1,2);
15:         return(a[1]+10);
16: }
17:
18: int     x,y;
19: main()
20: {
21:         x=f(100,200);
22:         y=g(x);
23: }
24: /*------------------------------------
25:         gcc -O -S  でコンパイル
26: -------------------------------------
27: _f:
28:         link a6,#0
29:         move.l 8(a6),d0 <= フレームポインタ相対
30:         add.l 12(a6),d0 <= フレームポインタ相対
31:         moveq.l #10,d1
32:         add.l d1,d0
33:         unlk a6
34:         rts
35: _g:
36:         link a6,#-12
37:         pea 2
38:         pea 1
39:         jsr _f
40:         move.l d0,-8(a6)
41:         moveq.l #10,d0
42:         add.l -8(a6),d0
43:         unlk a6
44:         rts
45: _main:
46:         link a6,#0
47:         pea 200
48:         pea 100
49:         jsr _f
50:         move.l d0,_x
51:         move.l d0,-(sp)
52:         jsr _g
53:         move.l d0,_y
54:         unlk a6
55:         rts
56: -------------------------------------
57:         gcc -O -S -fomit-frame-pointer でコンパイル
58: -------------------------------------
59: _f:                     <= link がない
60:         move.l 4(sp),d0 <= sp 相対
61:         add.l 8(sp),d0  <= sp 相対
62:         moveq.l #10,d1
63:         add.l d1,d0
64:         rts             <= unlk がない
65:         .even
66:         .globl _g
67: _g:
68:         link a6,#-12    <= ローカル変数がある場合
69:         pea 2
70:         pea 1
71:         jsr _f
72:         move.l d0,-8(a6)
73:         moveq.l #10,d0
74:         add.l -8(a6),d0
75:         unlk a6         <= link/unlk は省略できない
76:         rts
77: _main:                  <= link がない
78:         pea 200
79:         pea 100
80:         jsr _f
81:         move.l d0,_x
82:         move.l d0,-(sp)
83:         jsr _g
84:         move.l d0,_y
85:         add.w #12,sp
86:         rts             <= unlk がない
87: -------------------------------------*/
```

にやると大変な技術を要する。GCCでは大域変数のレジスタ割り付けはほんのオマケといった程度でしかなく，中途半端にレジスタ割り付けが行われるため性能が低下する場合のほうが多い。

ドライストンベンチマーク

いくらCコンパイラが理論的にいいコードを出すとしても実際のプログラムを動かしたときに性能が出なければ話になりません。ここでは一般的な（浮動小数点演算を含まない）プログラムの性質をよく表しているといわれているドライストンベンチマーク*8)によってGCCの性能を見ることにしましょう。使用するドライストンベンチマークのバージョンは2.1です。これは極端な最適化（関数のインライン展開）が行えないように改良されたドライストンベンチマークの（おそらく）最新バージョンです。コンパイル時のオプションはデフォルトで－O（最適化）を指定することにし，先に説明した，

－fstrength－reduce
－fomit－frame－pointer
－finline－functions

を適当に組み合わせてコンパイルを行い，性能の違いを測定しました。

表1に測定結果を示します。予想どおり関数のインライン展開は効果がありませんでした（バージョン2.1の性格）。しかし，スタックフレームの削除はかなり効果的であることがわかります。ドライストンベンチマークでは変数をregister宣言したときの値を測定するほうが普通ですが，GCCではregister宣言を行ったほうが性能が悪かったので，そちらは測定していません。これは無理矢理指定するregister宣言がGCC本来のレジスタ割り付けの邪魔をするせいであると考えられます。

一般的な10MHz動作の68000マシンでのドライストン値は900～1100程度ですからGCCはかなりよい結果を出しているといえるでしょう（ひとまず満足）。それにしてもXCの500という値は情けないですね。やはり次期バージョンに期待が高まります。

表1　ドライストンの結果（GCCバージョン 1.34を使用）

ドライストンベンチマーク　バージョン2.1の実行結果（OPMドライバあり）

コンパイルオプション（－O以外）			ドライストン値
—fomit—frame—pointer	—fstrength—reduce	—finline—functions	Dhrystone/sec
×	×	×	1204.8
○	×	×	1265.8
×	○	×	1190.5
○	○	×	1265.8
×	×	○	1162.8
○	×	○	1219.5
×	○	○	1204.8
○	○	○	1265.8

（注1）-finline-functions 使用時は Proc_8のコード生成を誤る。
ここでは手でパッチを当てて測定した。
（注2）GCCではレジスタ変数ありでコンパイルすると遅くなる。
（注3）同じ測定条件でのXCによる結果はレジスタ変数ありで 512.8 Dhrystone/sec,
レジスタ変数なしで 487.8 Dhrystone/sec。

＊8）ドライストン（dhrystone）のほかに有名なベンチマークテストとしては浮動小数点演算の性能を調べるウエットストン（whetstone），リアルタイム性能を調べるリアルストン（rhealstone），整数演算と浮動小数点演算の性能をペアで調べるスタンフォード（stanford）などがある。

参考文献

1) A.V. Aho and J.D. Ulman, "Principles of Compiler Design", Addison-Wesley, 1977（邦訳『コンパイラ』培風館，1986年）．

2) 松岡恭正，河内浩明，茂木強，「コンパイラにおける最適化の技法」，インターフェース，1989年3月号，pp. 196-210.

リスト10　関数のインライン展開

```
============== inline.c =================
 1: /*
 2:     （最適化９）関数のインライン展開
 3: */
 4: f(x,y)
 5: int x,y;
 6: {
 7:         return(x+y+10);
 8: }
 9:
10: g(x)
11: int x;
12: {
13:         int a[3];
14:         a[1]=f(1,2);
15:         return(a[1]+x);
16: }
17:
18: int     x,y;
19: main()
20: {
21:         x=f(100,200);
22:         y=g(x);
23: }
24: /*-------------------------------------
25:         gcc -O -S  でコンパイル
26: --------------------------------------
27: _f:
28:         link a6,#0
29:         move.l 8(a6),d0
30:         add.l 12(a6),d0
31:         moveq.l #10,d1
32:         add.l d1,d0
33:         unlk a6
34:         rts
35: _g:
36:         link a6,#-12
37:         pea 2
38:         pea 1
39:         jsr _f              <= f() を呼ぶ
40:         move.l d0,-8(a6)
41:         move.l 8(a6),d0
42:         add.l -8(a6),d0
43:         unlk a6
44:         rts
45: _main:
46:         link a6,#0
47:         pea 200
48:         pea 100
49:         jsr _f              <= f() を呼ぶ
50:         move.l d0,_x
51:         move.l d0,-(sp)
52:         jsr _g              <= g() を呼ぶ
53:         move.l d0,_y
54:         unlk a6
55:         rts
56: --------------------------------------
57:         gcc -O -S -finline-functions でコンパイル
58: --------------------------------------
59: _f:
60:         link a6,#0
61:         move.l 8(a6),d0
62:         add.l 12(a6),d0
63:         moveq.l #10,d1
64:         add.l d1,d0
65:         unlk a6
66:         rts
67: _g:
68:         link a6,#-12
69:         moveq.l #13,d1   <= f(1,2)=13 を展開
70:         move.l d1,-8(a6)<= a[1]=f(1,2)
71:         move.l 8(a6),d0 <= x
72:         add.l -8(a6),d0 <= x+a[1]
73:         unlk a6
74:         rts
75: _main:
76:         link a6,#-12
77:         move.l #310,d0   <= f(100,200)=310 を展開
78:         move.l d0,_x
79:         moveq.l #13,d1   <= f(1,2)=13 を展開
80:         move.l d1,-8(a6)<= a[1]=f(1,2)
81:         add.l -8(a6),d0 <= a[1]+x=g(x) を展開
82:         move.l d0,_y
83:         unlk a6
84:         rts
85: -------------------------------------*/
```

THE SENTINEL

●SLANG用アクションゲーム

今月のWORM KUNはSLANGで作られたアクションゲームです。ちょっと見るとどこにでもありそうなスネークゲーム。でもよく見ると，体が長くなること以外スネークゲームの様相はほとんど残されていません。

WORM君の大好物はRONちゃんです。RONちゃんを追っかけて，各面にいるものをすべて食べ尽くしてください。この場合，1匹ずつ地道にやっつけていってもよいのですが，リブルラブルの「バシシ」のように一定のエリアを囲んで一気に相手を飲み込むことも可能です。同じように見えるキャラクタにも実はいろいろ性格があるのがわかるでしょう。

さて，最初にいったようにこのゲームはSLANGで記述されています。SLANGは，比較的最近発表されたTTIやTTCに比べると，かなり本格的な開発言語だといえます。手軽に使うならTTI＆TTC，幅広く使うならSLANGという使い分けもできるでしょう。

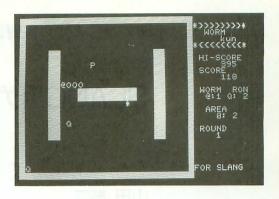
皆さんもこれらの言語を使って，アプリケーションをどんどん作ってみてください。

第86部
SLANG用ゲームWORM KUN
特別付録
再掲載　SLANGコンパイラ

●SLANG再掲載

S-OS用システム&アプリケーションのなかでも実用性の高いと思われるコンパイラ言語SLANGをファイル入出力ライブラリごと再掲載します。本当は先月のDIO.LIBと一緒に掲載する予定だったのですが，諸々の都合により今月の掲載となってしまいました。

今回再掲載されるのはコンパイラ本体だけですので，実際のプログラムの入力/作成にはS-OS用のエディタが必要です。具体的にいうと，主なところではZEDA，E-MATE，WINER，TED-750らのいずれかが必要となります。これら以外にもエディタ機能を持ったものがありますが，ZEDAコンパチのものならそのまま使用できます。また，エディタアセンブラREDAのエディタ部分はインデント（字下げ）を行わないという条件つきでなら使用できます。

SLANGは低級言語の小回りのよさと高級言語の便利さを兼ね備え，かつ手軽に扱えるコンパイラです。コンパイラによるプログラム開発の入門用とするなり，S-OS用のアプリケーションを開発するなり存分に活用してください。

●S-OSの系譜(6)

S-OSは現在のバージョン，“SWORD”になって格段の進化を遂げました。今月も先月に引き続いて，“SWORD”システム作成者のひとりである泉大介さんにS-OS“SWORD”誕生時の様子をうかがうことにしましょう。

S-OS“SWORD”のいちばんのウリはファイルシステムです。もともと，フロッピーディスクではディレクトリを一気に検索して目的のファイルが存在するかどうかをチェックします。一方，カセットテープでは，ファイルを順次読み込みながら目的のファイルかどうかチェックします。これら異なる動作をするものを，「ファイルオープン・アクセス・クローズ」という同じ方法でユーザーが使えるようにしようというのですから大変です。

これに対して私の友人がうまい解決方法を示してくれました。それは，Z80が持つ，キャリ/ゼロの2つのフラグを使い分けるという方法です。つまり，キャリフラグはメディアを問わずエラーの発生を意味させるようにします。ゼロフラグはディスクアクセスでは常にゼロであるように設定しておきます。一方，カセットテープでは目的のファイルが見つかればこれがゼロになるようにしておきます。このようにすることで「キャリフラグが立っていたらエラー，ノンゼロならゼロになるまでオープンし直す」というひとつのループで，テープにもディスクにも対応できるようになったのです。

また，これまでにS-OS“MACE”用に発表されていたプログラムがそのまま“SWORD”上で動くように用意された部分もあります。ファイルコントロールブロック，略してFCB（旧名RDI）がそうです。これは牛嶋さんによって提案，サポートされたルーチンで，ディスクをテープのように扱うものです。

このように“SWORD”には，“MACE”との互換性を維持するための工夫がいっぱいです。マシン語がわかるようになった皆さんは，一度見比べてみてください。

SLANG用ゲームWORM KUN

C言語ライクなS-OSオリジナルコンパイラSLANGで開発されたアクションゲームです。WORM君を操って画面上のRONちゃんをすべて食べ尽くしてください。なお，SLANGをお持ちでない方は145ページからのSLANG再掲載を参考にしてください。

Yamada Junji
山田 純二

1988年3月号で発表された構造型コンパイラSLANGを使ったリアルタイムゲームを発表します。

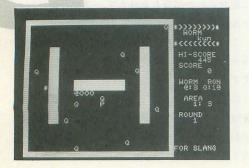
SLANGは，文法もCに近い構造型で機能もすっきりまとまっているし，浮動小数点パッケージを使えば実数演算までできてしまうS-OS用の開発ツールとしてはかなり強力なツールですが，いままでに発表されたものといえばユーティリティが数本，ゲームにいたっては1988年5月号に掲載された「地底最大の作戦」以外ない。これは実にもったいない状況といえるでしょう。

そこで，初めてSLANGを見る人向けに作ってみたのが今回のプログラムです。ゲームはシンプル（単に質素ともいう）そのものですが，これからSLANGを使ってみようかと思っている人は，今回のゲームプログラムを参考（踏み台？）にして，どんどんSLANGを活用してください。

入力方法

まず，掲載されているソースリストをE-MATEなどのエディタを使って入力し，ディスク版ユーザーの人はいったんデバイスにセーブしてからSLANGを起動し，

　]Cファイル名

でコンパイルしてください。テープ版ユーザーの人はオンメモリでコンパイルしなければならないので，エディタのテキスト格納アドレスを7000_H以降に変更してからテキストを入力しましょう。そしてコンパイルしてください。エラーなくコンパイルできた場合には，

　Program 8000～A3E2
　Work　　A3E3～????

と表示されますので，プログラムエリア8000_H～$A3E2_H$のオブジェクトをセーブして，J8000で実行してください。

操作法&遊び方

ゲーム内容はWORM君（“@”が頭で“O”が体です）を動かし，画面中にうろちょろしている“Q”（RONちゃんと呼んで）を10匹全部食べてしまうと，1面クリアというゲームです。育ち盛りのWORM君，1面クリアごとに体が1パーツずつ増えていき，最高15個分まで増え続けます。画面にはWORM君とRONちゃんのほかに，おじゃまキャラクターの“*”（アスタ君だよ）がいます。この“*”は，なにも考えずにただ画面中をうろちょろ進んでいるだけなのですが，WORM君にとっては天敵で体のどの部分に触れてもWORM君は死んでしまいます。

WORM君の移動は，カーソルキーか，テンキー，またはKを中心とした“I”，“M”，“J”，“L”で，上下左右に動かします。WORM君は，いったん進み始めたらほかの方向キーが押されるか，障害物にぶつかるまで進み続けます。それと自分の進んできた方向と逆向きには進めませんので注意してください。

次にスペースキーを1回押すと網を張るためのポインタ“P”を自分のいちばん後ろに置きます。“P”があるときにもう一度スペースキーを押すと，自分の頭“@”と“P”の範囲に網を張ることができます。自分で捕まえるほかに，この網を張ることでも“Q”を捕らえることができます。

画面右に“AREA”というのがありますが，これは網を張れる範囲を表示していてこれが0になってしまうと，一度置いたポインタは解除されてしまいます。このAREAも面クリアごとにひとつずつ増えていきます。RONちゃんは1匹捕らえると5点ですが，網を張って捕らえると倍の10点もらえ，複数いっぺんに捕まえるとスコアは2倍ずつ加算されます。

基本的にRONちゃんはランダムに動いていますがなかにはひねくれたヤツがいて，しつこくWORM君から逃げ回ります。こいつは，WORM君とX座標かY座標が合うと逃げてしまいますので，座標を少しずらして端に追い込んでからすばやく網を張って捕まえるのが効果的です。面が進むとだんだんひねくれたヤツが増えていき，10匹いっぺんに逃げるさまはなかなか気持ちいいです。

あと，プログラムの改造をしたいときに注意してほしいのは，このプログラム，ハッシュテーブルがめいっぱいなので，新しく関数を定義するときはSLANGのワークエリアの3012_Hと3014_Hからのハッシュ表の上限アドレスを変更してください。

最後に

初めに，初心者の人は参考にしてくださいなどといいましたが，コンパイル時に山のようなエラーが出てマニュアルとにらめっこしながらずいぶん苦労しました。プログラム自体の内容はたいしたことはないのに，なぜかずいぶんサイズが大きくなってしまいました（無駄が多いんでしょうね）。

このプログラムを作って，ひととおりSLANGを使ってみました。構造型ということで最初は戸惑っていましたが，使い慣れればプログラムはスッキリまとまるだろうし，使い方次第ではなんでもできちゃう魔法のSLANG！　となるでしょう。初心者の人にはとっつきにくいでしょうが，高級な処理からマシン語レベルの処理までこなせるこのSLANG，S-OS用の開発ツールとして，皆さんぜひ活用してください。

リスト1 WORM KUN

```
  1 //
  2 //  WORM KUN  1989.11.17-21
  3 //        FOR SLANG
  4 //     Programed by JUNJI
  5
  6         ORG   $8000;
  7
  8         VAR    LENG,I,J,SCORE,HISC,X,Y,
  9                DEATH,ROUND,PX,PY,PFLAG,
 10                HANI,NOKORI,RONALL,NAGASA,
 11                TKAZU,KCODE,SPC,APF,OVER;
 12
 13         ARRAY BYTE  WORM[31],
 14               BYTE  VRAM[29][24]:$B000,
 15               BYTE  RON[45],
 16               BYTE  BACK[300],
 17               BYTE  ASTA[15];
 18
 19 MAIN()
 20    VAR I,OVKEY;
 21  BEGIN
 22    WIDTH(40);
 23    REPEAT (
 24     SHOKI();
 25     GAMEN();
 26     MENSHOKI();
 27     PARAMETER();
 28     WORMPRINT();
 29     REPEAT (
 30     LOCATE(30,20); PRINT("PUSH ANY");
 31     LOCATE(30,21); PRINT("  KEY!");
 32     LOCATE(30,20); PRINT("        ");
 33     LOCATE(30,21); PRINT("        ");
 34     ) UNTIL INKEY(0)<>0;
 35      REPEAT (
 36       ASTAMAIN();
 37       FOR I=0 TO 2 (
 38        RONMAIN();
 39        WAIT();
 40        WORMMAIN();
 41        IF PFLAG<>0 THEN POINTPUT();
 42        IF RONALL==0 THEN ROUNDCLEAR();
 43        IF DEATH<>0 THEN (
 44           WSHINDA();I=0;
 45         ) ENDIF;
 46       ) NEXT;
 47       APF--;
 48      )
 49      UNTIL OVER<>0;
 50      OVKEY=GAMEOVER();
 51     ) UNTIL OVKEY==2;
 52    LOCATE(0,0);
 53  END;
 54
 55 WORMMAIN()
 56    VAR  KEY,FLAG;
 57 BEGIN
 58      KEY=KEYSET();
 59      IF KEY==0 THEN KEY=KCODE;
 60         ELSE IF KEY<>5 THEN KCODE=KEY;
 61       X=WORM[0];
 62       Y=WORM[1];
 63       CASE KEY OF (
 64          1   Y--;
 65          2   X++;
 66          3   Y++;
 67          4   X--;
 68          5   IF SPC==0 THEN (;
 69               POINTSET();SPC=10;
 70             ) ENDIF;
 71          10   OVER=10;
 72       )
 73     IF KEY<>0 THEN (
 74       FLAG=XYCHK();
 75       FLAG=CHRCHK();
 76        IF FLAG==0 THEN (
 77          WORMMOVE();
 78       ) ENDIF;
 79     )  ENDIF;
 80      WORMPRINT();
 81 END;
 82
 83 KEYSET()
 84    VAR CODE;
 85  BEGIN
 86     CODE=INKEY(0);
 87      CASE CODE OF (
 88              '2'  CODE=3;
 89              '4'  CODE=4;
 90              '6'  CODE=2;
 91              '8'  CODE=1;
 92              'M'  CODE=3;
 93              'J'  CODE=4;
 94              'L'  CODE=2;
 95              'I'  CODE=1;
 96              $1F  CODE=3;
 97              $1D  CODE=4;
 98              $1C  CODE=2;
 99              $1E  CODE=1;
100              ' '  CODE=5;
101              $1B  CODE=10;
102           OTHERS CODE=0;
103         )
104       IF CODE<>5 THEN ( SPC=0;
105          IF CODE<>0 THEN CODE=TURNCHK(CODE);
106     ) ENDIF;
107  END(CODE);
108
109 TURNCHK(CODE)
110   VAR   CD;
111   BEGIN
112    CD=CODE;
113     CODE=CODE+2;
114      IF CODE>4 THEN CODE=CODE-4;
115      IF KCODE==CODE THEN CD=0;
116     END(CD);
117
118 WORMMOVE()
119     VAR  XT,YT;
120   BEGIN
121     I=LENG*2-2;
122     XT=WORM[I];
123     YT=WORM[I+1];
124     LOCATE(XT,YT); PRINT(" ");
125       REPEAT (
126         WORM[I]=WORM[I-2];
127         WORM[I+1]=WORM[I-1];
128         I=I-2;
129     ) UNTIL I<2;
130      WORM[0]=X;WORM[1]=Y;
131    END;
132
133 XYCHK()
134     VAR CHK;
135   BEGIN
136      CHK=0;
137      IF X==0  THEN CHK=10;
138      IF X==29 THEN CHK=10;
139      IF Y==0  THEN CHK=10;
140      IF Y==29 THEN CHK=10;
141   END(CHK);
142
143 CHRCHK()
144     VAR CHK,DISP;
145   BEGIN
146     CHK=0;
147       DISP=SCREEN(X,Y);
148         CASE DISP OF (
149            $7B  CHK=10;
150            'Q'  ( GETRON();
151                      SCOREPLUS(5); )
152              '*'   DEATH=10;
153          )
154   END(CHK);
155
156 WSHINDA()
157   BEGIN
158         WORMLINE('*');
159           BEEP(); BEEP();
160             WAIT();
161             WORMLINE(' ');
162             NOKORI--;
163         IF NOKORI==0 THEN OVER=10;
164           LOCATE(PX,PY);
165           PRINT(" ");
166           PFLAG=0;
167         DEATH=0;
168        PARAMETER();
169       WORMSET();
170       ASTAOFF();
171       ASTACLR();
172      APF=1;
173    END;
174
175 ASTAOFF()
176   BEGIN
177    FOR I=0 TO 4 (
178     IF ASTA[I*3]<>0 THEN (
179         LOCATE(ASTA[3*I+1],ASTA[I*3+2]);
180         PRINT(" ");
181      ) ENDIF;
182    ) NEXT;
183    END;
184
```

THE SENTINEL

```
185 WORMLINE(CH)
186 `BEGIN
187    FOR I=0 TO LENG-1 (
188      LOCATE(WORM[I*2],WORM[I*2+1]);
189       PRINT(CHR$(CH));
190       ) NEXT;
191   END;
192
193
194 RONDEATH(XR,YR)
195    VAR  RI,I;
196  BEGIN
197   RI=99; I=0;
198    REPEAT (
199     IF RON[I]<>0 THEN (
200          IF RON[I+1]==XR THEN (
201           IF  RON[I+2]==YR THEN RI=I;
202       ) ENDIF;
203    ) ENDIF;
204     I=I+3;
205       ) UNTIL I>44;
206    RON[RI]=0;
207    END(RI);
208
209 GETRON()
210    VAR  RI;
211  BEGIN
212    RONDEATH(X,Y);
213     IF RI<>99 THEN (
214          LOCATE(X,Y);
215           PRINT(" ");
216         RONALL--; PARAMETER();
217     ) ENDIF;
218    BEEP();
219    END;
220
221 WORMPRINT()
222  BEGIN
223   I=LENG*2-2;
224     REPEAT (
225      LOCATE(WORM[I],WORM[I+1]);
226      PRINT("O");I=I-2;
227    ) UNTIL I<2;
228     LOCATE(WORM[0],WORM[1]);
229     PRINT("@");
230  END;
231
232
233 POINTSET()
234  BEGIN
235    IF PFLAG==0 THEN (
236          PFLAG=10;
237            PX=WORM[LENG*2-2];
238            PY=WORM[LENG*2-1];
239             POINTPUT(); BEEP();
240          ) ELSE (
241            LOCATE(PX,PY);PRINT(" ");
242            PFLAG=0;
243          BOXFULL(PX,PY);
244        LOCATE(33,15);
245        PRINT(FORM$(NAGASA,02),":",FORM$(NAGASA,02));
246           )
247     ENDIF;
248  END;
249
250 POINTPUT()
251    VAR  PF;
252  BEGIN
253   LOCATE(PX,PY);
254     PRINT("P");
255      PF=POINTCHK();
256      IF PF<>0 THEN (
257         PFLAG=0; BEEP();
258         LOCATE(PX,PY);
259       PRINT(" ");
260        LOCATE(33,15);
261         PRINT(FORM$(NAGASA,02),":",FORM$(NAGASA,02));
262     ) ENDIF;
263   END;
264
265 POINTCHK()
266      VAR    SX,SY,PF;
267  BEGIN
268    PF=0;
269    SX=PX-WORM[0];
270    SY=PY-WORM[1];
271      IF SX.<.0 THEN (
272          SX=CPL(SX)+1;
273        ) ENDIF;
274      IF SY.<.0 THEN (
275          SY=CPL(SY)+1;
276        ) ENDIF;
277      IF  NAGASA<SX THEN PF=10;
278      IF  NAGASA<SY THEN PF=10;
279    LOCATE(33,15);
280    PRINT(FORM$(NAGASA-SX,02),":",FORM$(NAGASA-SY,2));
281      END(PF);
282
283 BOXFULL(RX,RY)
284      VAR  LX,LY,SC,CN;
285   BEGIN
286     LX=WORM[0];
287      LY=WORM[1];
288       IF LX>RX THEN (
289         I=RX;RX=LX;LX=I;
290         ) ENDIF;
291          IF LY>RY THEN (
292         I=RY;RY=LY;LY=I;
293         ) ENDIF;
294           BACKCHK(RX,RY,LX,LY);
295           FOR I=LY TO RY (
296           LOCATE(LX,I);
297           PRINT(STR$($7B,RX-LX+1));
298         ) NEXT;
299           IF TKAZU<>0 THEN (
300            TDEATHPRT();
301              SC=0; CN=10;
302                FOR I=0 TO TKAZU-1 (
303                  SC=SC+CN; CN=CN*2;
304                ) NEXT;
305         ) ENDIF;
306           WAIT(); SCOREPLUS(SC);
307           FOR I=LY TO RY (
308           FOR J=LX TO RX (
309          LOCATE(J,I);
310          PRINT(CHR$(VRAM[J][I]));
311       ) NEXT;
312     ) NEXT;
313   END;
314
315 BACKCHK(BRX,BRY,BLX,BLY)
316      VAR   CH;
317   BEGIN
318      TKAZU=0;
319       FOR I=BLY TO BRY (
320        FOR J=BLX TO BRX (
321          CH=SCREEN(J,I);
322          CASE CH OF (
323             'Q' (
324                   RONALL--;
325                     PARAMETER();
326                   RONDEATH(J,I);
327                 TBSET(J,I,CH);
328           ) )
329         ) NEXT;
330      ) NEXT;
331
332     END;
333
334 TBSET(HX,HY,CH)
335   BEGIN
336     BACK[TKAZU*3]=CH;
337     BACK[TKAZU*3+1]=HX;
338     BACK[TKAZU*3+2]=HY;
339     TKAZU++;
340     END;
341
342
343 TDEATHPRT()
344    VAR RI,XC,YC;
345   BEGIN
346      FOR I=0 TO TKAZU-1 (
347       XC=BACK[I*3+1];
348       YC=BACK[I*3+2];
349        CASE BACK[I*3] OF (
350          'Q' ( RI=RONDEATH(XC,YC);
351              )
352           )
353         LOCATE(XC,YC);PRINT("*");
354       BEEP();
355      ) NEXT;
356     END;
357
358 //     ENEMY MAIN
359
360 RONMAIN()
361   BEGIN
362    I=0;
363     REPEAT (
364       IF RON[I]<>0 THEN RONMOVE();
365      I=I+3;
366    )  UNTIL I>43;
367     END;
368
369 RONMOVE()
370      VAR    RNX,RNY,RN;
371   BEGIN
372    RNX=RON[I+1];
373    RNY=RON[I+2];
374     LOCATE(RNX,RNY);
375     PRINT(" ");
376      IF RON[I]<>1 THEN (
```


▶めぞん一刻24枚組LD購入のためX68000が買えなくなってしまったという方はけっこういると思いますが，春ごろにあの伝説の50枚組LD「うる星やつら」が再販されるそうです。またX68000が遅くなる？　LDプレーヤーも安くなったことだし。さあ皆さん，30万円どうやって作ります!?
田中　正志(21)千葉県

```
377         RN=RNIGERU(RNX,RNY);
378       ) ELSE  ( RN=RND(4);
379     ) ENDIF;
380     CASE RN OF (
381        0  RNY++;
382        1  RNX--;
383        2  RNX++;
384        3  RNY--;
385   )
386      IF SCREEN(RNX,RNY)==$20 THEN (
387             RON[I+1]=RNX;
388             RON[I+2]=RNY;
389       ) ENDIF;
390        LOCATE(RON[I+1],RON[I+2]);
391      PRINT("Q");
392  END;
393
394 RNIGERU(RNX,RNY)
395    VAR   RDX,RDY,RN,CH;
396  BEGIN
397   RDX=RNX-WORM[0];
398   RDY=RNY-WORM[1];
399
400     IF RDX==0 THEN (
401         IF RNX<14 THEN RN=2;
402           ELSE RN=1;
403       ) ENDIF;
404     IF RDX.<.0 THEN RN=1;
405     IF RDX.>.0 THEN RN=2;
406       CH=RCHK(RNX,RNY,RN);
407        IF CH<>$20 THEN (
408         IF RDY==0 THEN (
409             IF RNY<11 THEN RN=0;
410                 ELSE RN=3;
411           ) ENDIF;
412         IF RDY.<.0 THEN RN=3;
413         IF RDY.>.0 THEN RN=0;
414     ) ENDIF;
415   END(RN);
416
417 RCHK(RNX,RNY,RN)
418   VAR   CH;
419  BEGIN
420    CASE RN OF (
421       0  RNY++;
422       1  RNX--;
423       2  RNX++;
424       3  RNY--;
425      )
426      CH=SCREEN(RNX,RNY);
427    END(CH);
428
429 ASTAMAIN()
430  BEGIN
431    FOR I=0 TO 4 (
432       IF ASTA[I*3]<>0 ASTAMOVE();
433           ELSE ASTADASU();
434     ) NEXT;
435    END;
436
437 ASTADASU()
438  BEGIN
439   IF APF==0 THEN (
440     APF=100-ROUND*10;
441      IF APF<5 THEN APF=5;
442      ASTA[I*3]=RND(02)+1;
443      REPEAT (
444        ASTA[I*3+1]=RND(28)+1;
445        ASTA[I*3+2]=RND(2)*22+1;
446    ) UNTIL SCREEN(ASTA[I*3+1],ASTA[I*3+2])==$20;
447     LOCATE(ASTA[I*3+1],ASTA[I*3+2]);
448     PRINT("*"); BEEP();
449   ) ENDIF;
450   END;
451
452 ASTAMOVE()
453   VAR   XA,YA,DX,DY,DFL,FC;
454  BEGIN
455   XA=ASTA[I*3+1];
456   YA=ASTA[I*3+2];
457    LOCATE(XA,YA);
458    PRINT(" ");
459    CASE ASTA[I*3] OF (
460        01  DX=-1, DY=1;
461        02  DX=1 , DY=1;
462        03  DX=1 , DY=-1;
463        04  DX=-1, DY=-1;
464      )
465      DFL=0;
466      FC=ASTACHK(XA+DX,YA);
467       IF FC<>0 THEN (
468         DX=CPL(DX)+1; DFL=10;
469        ) ENDIF;
470      FC=ASTACHK(XA+DX,YA+DY);
471      IF  FC<>0 THEN (
472            DY=CPL(DY)+1; DFL=10;
473          ) ENDIF;
474       IF DFL<>0 THEN (
475           ASTA[I*3]=DIRSET(DX,DY);
476         ) ELSE (
477        ASTA[I*3+1]=ASTA[I*3+1]+DX;
478        ASTA[I*3+2]=ASTA[I*3+2]+DY;
479      ) ENDIF;
480    LOCATE(ASTA[I*3+1],ASTA[I*3+2]);
481     PRINT("*");
482    END;
483
484 ASTACHK(ASX,ASY)
485   VAR  AW;
486  BEGIN
487   AW=SCREEN(ASX,ASY);
488     CASE AW OF (
489      'O'  DEATH=10, AW=0;
490      '@'  DEATH=10, AW=0;
491      $7B  AW=10;
492    OTHERS AW=00;
493      )
494   END(AW);
495
496 DIRSET(DX,DY)
497   VAR  DIR;
498 BEGIN
499   CASE DX*3+DY OF (
500      -2  DIR=1;
501       4  DIR=2;
502       2  DIR=3;
503      -4  DIR=4;
504     )
505   END(DIR);
506
507 ROUNDCLEAR()
508   VAR  K;
509  BEGIN
510    LOCATE(05,12);
511     PRINT("**  ROUND CLEAR!! **");
512     BEEP();
513     ROUND++;LENG++;NAGASA++;
514      IF LENG==16 THEN LENG=15;
515      IF NAGASA==16 THEN NAGASA=15;
516     LOCATE(07,14);
517      PRINT("PUSH RETURN KEY!");
518       REPEAT (
519        K=INKEY(2);
520       ) UNTIL K==$0D;
521     VRAMCLR();
522     GAMEN();
523    MENSHOKI();
524    PARAMETER();
525   WORMPRINT();
526   KCODE=1;
527  END;
528
529 GAMEOVER()
530   VAR    K;
531  BEGIN
532    LOCATE(08,12);
533     PRINT("** GAME OVER **");
534    LOCATE(08,14);
535      PRINT("TRY AGAIN(Y/N)?");
536      REPEAT (
537       K=INKEY(2);
538       CASE K OF (
539         'Y'  K=1;
540         'N'  K=2;
541      OTHERS K=0;
542      )
543       ) UNTIL K<>0;
544    END(K);
545
546 MENSHOKI()
547  BEGIN
548     APF=1;
549     PFLAG=0;
550     KCODE=1;
551      WORMSET();
552      RONSET();
553      ASTACLR();
554   END;
555
556 WORMSET()
557  BEGIN
558      I=0;
559      REPEAT (
560        WORM[I]=15;
561         WORM[I+1]=15;
562        I=I+2;
563    ) UNTIL I>LENG*2-2;
564   END;
565
566 ASTACLR()
567  BEGIN
568    FOR I=0 TO 15  (
```

▶僕の周りには毎日学校帰りにゲームやってて小学生の間で有名になっているやつとか,ゲーセンに行って有り金使ってくると学校に出てくるやつとか,2日ゲームをしないと指がけいれんするやつとかいて,ふと冬の青空を見たりすると「そーいや僕ら受験生だ」と思い出す今日このごろです。
荻島　満(18)静岡県

```
569       ASTA[I]=0;
570     ) NEXT;
571   END;
572
573 RONSET()
574    VAR   TC,NIGE;
575   BEGIN
576      NIGE=ROUND MOD 10;
577      FOR I=0 TO 09 (
578        IF NIGE<>0 THEN (
579        RON[I*3]=2; NIGE--;
580        ) ELSE (
581          RON[I*3]=1;
582        ) ENDIF;
583          REPEAT (
584            X=RND(29)+1;
585            Y=RND(22)+1;
586          TC=SCREEN(X,Y);
587        ) UNTIL TC==$20;
588          RON[I*3+1]=X;
589          RON[I*3+2]=Y;
590        LOCATE(X,Y);PRINT("Q");
591      ) NEXT;
592    RONALL=10;
593   END;
594
595 SHOKI()
596   BEGIN
597      SCORE=0;
598      ROUND=1;
599      NOKORI=3;
600      NAGASA=5;
601       LENG=4;
602       DEATH=0;
603       OVER=0;
604    VRAMCLR();
605   END;
606
607 VRAMCLR()
608   BEGIN
609     FOR I=0 TO 24 (
610       FOR J=0 TO 29 (
611         VRAM[J][I]=$20;
612       ) NEXT;
613     ) NEXT;
614   END;
615
616 GAMEN()
617    BEGIN
618     PRINT(CHR$($0C));
619       PRINT(STR$($7B,30));
620       LOCATE(0,1);
621        FOR I=1 TO 23 (
622         PRINT(CHR$($7B),SPC$(28),CHR$($7B),"¥N");
623         VRAM[0][I]=$7B;
624         VRAM[29][I]=$7B;
625         ) NEXT;
626       PRINT(STR$($7B,30));
627      STAGEPRINT();
628     END;
629
630 PARAMETER()
631   BEGIN
632       LOCATE(30,1);
633       PRINT("*>>>>>>>>*");
634        LOCATE(30,2);
635        PRINT("  WORM");
636          LOCATE(30,3);
637          PRINT("      kun");
638           LOCATE(30,4);
639           PRINT("*<<<<<<<<*");
640           SCOREPRINT();
641          LOCATE(31,11);
642         PRINT("WORM  RON");
643       LOCATE(32,12);
644      PRINT("@:",NOKORI," Q:",FORM$(RONALL,2));
645      LOCATE(32,14);
646      PRINT("AREA");
647      LOCATE(33,15);
648     PRINT(FORM$(NAGASA,02),":",FORM$(NAGASA,02));
649     LOCATE(31,17);
650     PRINT("ROUND");
651     LOCATE(33,18);
652    PRINT(FORM$(ROUND,02));
653    LOCATE(30,23);
654    PRINT("FOR SLANG");
655   END;
656
657 SCOREPLUS(SC)
658   BEGIN
659    SCORE=SCORE+SC;
660     IF SCORE>HISC THEN HISC=SCORE;
661     SCOREPRINT();
662    END;
663
664 SCOREPRINT()
665   BEGIN
666     LOCATE(31,6);
667       PRINT("HI-SCORE");
668       LOCATE(33,7);
669       PRINT(%(HISC));
670         LOCATE(31,8);
671         PRINT("SCORE");
672         LOCATE(33,9);
673        PRINT(%(SCORE));
674   END;
675
676 STAGEPRINT()
677    VAR  SA;
678   BEGIN
679     SA=ROUND MOD 5;
680      CASE SA OF (
681            1    STAGE1();
682            2    STAGE2();
683            3    STAGE3();
684            4    STAGE4();
685            0    STAGE0();
686          )
687     END;
688
689 STAGE1()
690   BEGIN
691       BLOCKPRINT(5,5,6,0,2);
692       BLOCKPRINT(23,5,6,0,2);
693       BLOCKPRINT(10,11,4,2,0);
694     END;
695
696 STAGE2()
697   BEGIN
698       BLOCKPRINT(7,5,6,0,2);
699       BLOCKPRINT(21,5,6,0,2);
700       BLOCKPRINT(5,11,9,2,0);
701     END;
702
703 STAGE3()
704   BEGIN
705       BLOCKPRINT(6,5,6,0,2);
706       BLOCKPRINT(22,5,6,0,2);
707       BLOCKPRINT(14,1,3,0,2);
708       BLOCKPRINT(14,17,3,0,2);
709     END;
710
711 STAGE4()
712   BEGIN
713       BLOCKPRINT(7,5,6,0,2);
714       BLOCKPRINT(21,5,6,0,2);
715       BLOCKPRINT(5,19,9,2,0);
716     END;
717
718 STAGE0()
719   BEGIN
720       BLOCKPRINT(5,5,9,2,0);
721       BLOCKPRINT(10,1,6,0,2);
722       BLOCKPRINT(5,19,9,2,0);
723     END;
724
725 BLOCKPRINT(X,Y,N,DX,DY)
726     VAR I;
727   BEGIN
728     FOR I=0 TO N (
729         BLOCKPUT(X,Y);
730           X=X+DX; Y=Y+DY;
731     ) NEXT;
732 END;
733
734 BLOCKPUT(XB,YB)
735     VAR  XS;
736   BEGIN
737      LOCATE(XB,YB);
738       PRINT(STR$($7B,2),"¥D¥L¥L",STR$($7B,2));
739         FOR I=0 TO 1 (
740           XS=XB;
741         FOR J=0 TO 1 (
742            VRAM[XB][YB]=$7B;
743           XB++;
744         ) NEXT;
745       XB=XS;
746       YB++;
747      ) NEXT;
748   END;
749
750 WAIT()
751     VAR  MATU;
752   BEGIN
753     MATU=2800-(200*(ROUND MOD 10)+1);
754      FOR I=0 TO MATU (
755        ) NEXT;
756   END;
```

全機種共通S-OS“SWORD”要

再掲載SLANGコンパイラ

1988年3月号で発表された，S-OSオリジナルコンパイラ言語SLANGを再掲載します。小回りがききながら極端な癖のない構造化言語ですから，常用プログラミングツールとして活用できます。また，デバッグ情報としても活用してください。

SLANGは1988年3月号に掲載された構造型コンパイラ言語です。命令などを見てもわかるようにC言語にかなり近い文法を持っており，さらにマシン語レベルの低級処理，プリプロセッサまがいのコンパイラ命令を備えた高機能言語です。

その後，実数演算パッケージとのリンク，ファイル入出力機能の追加，さらに12月号で行われたリダイレクション機能のサポートによりS-OSに高機能コマンドを追加することもできるようになりました。

S-OS用ユーティリティの開発やホビープログラミングだけでなく，プログラミングの学習用，C言語への入門用と，さまざまな用途に適した手ごろな言語だと思います。

今回再掲載されるのは，バグ取りおよびファイル入出力拡張（デバッグ版）を加えたものです。今回のデバッグを行わないとディスクを破壊することがありますので，これまでのSLANGユーザーの方も必ず確認しておいてください。

また，従来は仕様書どおりの動作でなかったSGN( )関数も仕様書どおりに訂正されました。

SLANGの言語仕様

SLANGは2バイト無符号整数を基本としたコンパイラ言語です。1パスでコンパイルを終了するので非常に高速な開発環境が実現できます。

C言語同様プログラムはmain( )から実行されます。かなりの部分がC言語を意識した構成になっていますが，Z80にとって重すぎる部分などは簡略化されたり，さらにZ80に最適化された仕様も加えられています。I/Oエリアを配列にしたり，S-OSの特殊ワークをサポートしたり，レジスタに密着した変数を用意したりといった部分がそれにあたります。マシン語の知識があればそれを十二分に生かせる言語といえるでしょう。

C言語のようなポインタは使えませんが，間接変数によって「ポインタもどき」くらいならできますし，局所変数サポート，配列も2次元まで扱えます。

通常，オブジェクトにはランタイムルーチンが加えられていますので，できあがったオブジェクトの先頭番地から実行すればそのままプログラムを起動できます。

使い方

さて，SLANGはオンメモリ版とディスク版の2とおりの使い方ができます。テープ，QDユーザーの方は3006H番地を0に書き換えたオンメモリ版を使用します。この場合はあらかじめエディタで一定のアドレスからソースを書き込んでおく必要があります。そのほかの方はそのままディスク版をお使いください。

SLANGをメモリ上に読み込み，

J3000

でコールドスタートです。この状態で以下のコマンドが使用できます。

Cファイル名

ファイルを読み込みコンパイルを開始します。オンメモリ版のときはメモリ上のソースをコンパイルします。

X　nn

ソース格納アドレスをnn番地に変更します。

S nn1 nn2 nn3 nn4：ファイル名

読み込んだときnn1からnn2番地にロードされnn3番地から実行されるようなオブジェクトをセーブします。nn4はセーブするプログラムの実際の格納アドレスでnn1と同じ場合は省略できます。

D　デバイス名：

ディレクトリを表示します。

DV　デバイス名：

デフォルトデバイスを変更します。

#

プリンタをON/OFFします。

JまたはG　nn

nn番地のオブジェクトを実行します。

!

S-OSのモニタに移ります

M

各機種のモニタに移ります。

表1　初期値およびスイッチ

$3006：DISK　　初期値…1
　0：オンメモリ版
　1：ディスク版
$3007：セミコロン　チェック　初期値…1
　0：チェックしない
　1：チェックする
$3008：OBJ初期値（下位）　初期値…$B000
$3009：OBJ初期値（上位）
　オブジェクトコードを生成する先頭アドレス。ORG宣言を省略した場合使用される。ディスク版の場合$8000にするとよい
$300A：ランタイム最終ADR（下位）
$300B：　　〃　　（上位）
　ランタイムルーチンの最終アドレス。ランタイムルーチンを追加する場合変更する
$300C：クラスタBUFF(下位)　初期値…$7000
$300D：　　〃　　（上位）
　ディスク版で，ソースを読み込むための4Kバイトのワークの先頭アドレス
$300E：TEXT. TOP（下位）　初期値…$7000
$300F：TEXT. TOP（上位）
　オンメモリ版で，ソースを格納する先頭アドレスの初期値。Xコマンドで変更することができる
$3010：局所表TOP（下位）　初期値…$0000
$3011：　　〃　　（上位）
　S-OS特殊ワークエリア上に取られる局所表の先頭アドレス。通常変更しない
$3012：ハッシュ表TOP(下位)　初期値…$0200
$3013：　　〃　　（上位）
　S-OS特殊ワークエリア上に取られる大域表のためのハッシュ表の先頭アドレス。同時に局所表の上限でもある
$3014：大域表TOP（下位）　初期値…$0300
$3015：　　〃　　（上位）
　S-OS特殊ワークエリア上に取られる大域表の先頭アドレス。同時にハッシュ表の上限でもある

表2　SLANGリファレンスマニュアル

書式に関する規定

フリーフォーマット

基本的にはC言語のようにフリーフォーマットで，行の概念はなく，名前などの途中以外ではどこで区切ってもよいがいくつか例外がある。

//コメント　//以降はコメントと見なされ，その行の終わりまで無視される

"文字列"　2行にまたがることはできない

配列　配列名と[の間を空白などで区切ることはできない

関数　関数名と(の間を空白などで区切ることはできない

空白

空白は名前などの途中と配列の[および関数の(の前以外ならどこに置いてもよい。以下のものは空白と同等である。

改行
コメント
#コマンド

コメント

注釈文。空白が置けるところなら，どこに置いてもよい。

//コメント　//から行の終わりまで

/*コメント*/　/*から*/まで。ネスティング不可

(*コメント*)　(*から*)まで。ネスティング不可

#コマンド

コンパイラに対する命令。空白が置けるところなら，どこに置いてもよい。

#INCLUDE　ファイルネーム

別のソースをその場所に取り込む。ネスティング不可。ファイルネーム以降は行の終わりまで無視される。オンメモリ版では使用できない

#CHAIN　ファイルネーム

続きのソースを読み込む。ファイルネーム以降はすべて無視される。オンメモリ版の場合，準備がよいかどうか聞いてくるので，なにかキーを押すと読み込みを始める。ブレイクキーを押すとコンパイルを中止する

#IF 式
#ELSE
#ENDIF

条件付きコンパイルを行う。#IFの後ろの式が真ならば#ELSEまでを，偽ならば#ELSEから#ENDIFまでをコンパイルする

アドレス宣言

オブジェクトコードやワークエリアの先頭アドレスを指定する。宣言を省略することもできる。

ORG宣言

オブジェクトコードの先頭アドレスを指定する。宣言を省略した場合はデフォルト値が使われる。実際には，先頭にランタイムルーチンがリロケートされ，その後ろにオブジェクトコードが続く

WORK宣言

変数や配列のワークエリアの先頭アドレスを指定する。まず静的なワークエリアが取られ，その後ろから$FFFFに向かって，動的なワークエリアが伸びていく

宣言を省略した場合は，静的なワークエリアはオブジェクトコード中に埋め込まれ，動的なワークエリアはオブジェクトコードの後ろから$FFFFに向かって伸びていく。ただし，初期値を持つ静的なワークエリアの場合は，宣言の有無にかかわらず，オブジェクトコード中に埋め込まれる

OFFSET宣言

コードを生成する際のオフセットを指定する。ZEDAやFuzzyBASICコンパイラのOFFSETと同じ。

宣言を省略した場合はオフセットは0となる

プログラム

アドレス宣言，大域宣言とブロック（関数定義）からなる。必ず，

MAIN( )

という関数が必要で，プログラムを実行させることは関数MAIN( )を実行させることである。関数MAIN( )の定義はプログラムのどこにあってもよい。

ブロック

関数頭書き，局所的宣言（静的宣言と局所宣言）と関数定義からなる。

```
SUB(X, Y)            (*関数頭書き*)
  VAR I;             (*静的宣言*)
  BEGIN              (*関数定義*)
    I=X+Y;
    RETURN(I);
  END;
```

局所的なまとまりで，この中で宣言された名前はこの中でのみ有効となる。

名前の有効範囲

局所的な名前

静的宣言や局所宣言で宣言された名前や仮引数，ラベル名は局所的な名前となり，その関数内でのみ使用できる。大域的な名前に同じ名前があった場合，局所的な名前を優先する

大域的な名前

関数名や大域宣言で宣言された名前は大域的な名前となり，プログラム全体で使用できる。関数名以外は，宣言された以後有効となる

大域宣言

大域的な名前を宣言する。変数や配列は静的にメモリに割り付ける。

静的宣言

局所的な名前を宣言する。変数や配列は静的にメモリに割り付ける。

局所宣言

局所的な名前を宣言する。変数や配列は動的に取る。ただし，そのワークエリアの合計は1関数240バイト以内でなければならない。

型

BYTE, !　…1バイト型
WORD, %…2バイト型

データ形式

変数

符号なし16ビット長整数を扱い，型はない。単純変数と間接変数があり，必ずVAR宣言が仮引数で宣言してから使用する。

間接変数は，変数としても，配列としても扱える。FuzzyBASICの変数，メモリ配列の扱いと同じ。間接変数自体に型はないが，配列として使用するため，1バイト型か2バイト型かを宣言しなければならない。省略した場合は，2バイト型と見なされる。

たとえば，POINTが2バイト型の間接変数として宣言されていたとすると，

POINT=$C000;
I=POINT[3];

では，$C006の内容を下位バイト，$C007の内容を上位バイトとして，変数Iに代入される。

単純変数は変数としてのみ使用でき，配列としては扱えない。

VAR宣言

変数を宣言する。仮引数リストも書式は同じ。複数の変数を宣言する場合には，(カンマ)でつなぐ。

単純変数は

VAR HENSUU, ABC;

のように変数名を書けばよい。

間接変数は配列として使用する際の型を宣言し，変数名に[ ]を付けた形で宣言する。

VAR BYTE POINT [ ], %KANSETU[ ];

型を省略した場合は2バイト型と見なされる。

後ろに：定数式とすることにより，変数の格納アドレスを指定することができる。

VAR XY:$C000, BYTE Z[ ]:$D000;

と書くと，$C000と$C001を変数XYの格納アドレス，$D000と$D001を変数Zの格納アドレスとすることができる。

また，=定数式とすることにより，変数の初期化をすることができる。この場合，WORK宣言がなされていても，その変数のワークはプログラム中に埋め込まれる。

VAR A=0, B=3, C[ ]=$C000;

この初期化はコンパイル時にのみ行われ，実行時には行われない。

ただし，変数の格納アドレス指定と初期化は大域宣言と静的宣言のみ使用でき，局所宣言や仮引数リストでは使用できない。

定数

基本的には，16ビット長の符号なし整数で0から65535までの値を取るが，2の補数表現の符号付きの整数と見ることができる。

10進数

数字からなる文字列
例) 1234, −5

16進数

$で始まり16進数からなる文字列か数字で始まり，かつ16進数からなり最後にHが付く文字列
例) $ABCD, 12ABH, 0FFFFH

2進数

0と1からなる文字列で，最後にBが付く
例) 1111000011001010B

文字定数

'(シングルクォーテーション)でくくった1文字で，文字のASCIIコードを値とする。エスケープ文字が使用可
例) 'A', '\N', '\'

文字列定数

"(ダブルクォーテーション)でくくった文字列で，文字列が格納されているアドレスを値とする。文字列は，オブジェクトコード中に埋め込まれ，自動的に最後に$00が付けられる。エスケープ文字が使用可。

ただし，2行にまたがることはできず，定数式にも使用できない
例) "メッセージ\n"

記号定数

CONST宣言で定義された値を持つ

$

次に生成するオブジェクトコードのアドレスを値とする

エスケープ文字

文字定数や文字列定数中に使われ，2文字で1文字として扱われる。\の後ろに1文字を付けた形で使用されるが，該当する文字がない場合は\だけで1文字となる。大文字と小文字の区別はしない。

\\ ……\
\" ……"
\' ……'
\N ……$0D
\/ ……$0D
\C ……$0C
\R ……$1C
\L ……$1D
\U ……$1E
\D ……$1F
\0 ……$00　(MZ以外では\は¥)

CONST宣言

記号定数を定義する。複数の記号定数を定義する場合は，(カンマ)でつなぐ。

CONST PC=$8001, MZ=2000;

とすると，以後PCは$8001，MZは2000という定数値を

持つ。
CONST宣言は静的宣言と局所宣言の差異はなく，どちらも局所的な記号定数の宣言となる。

配列

1バイト型と2バイト型がある。単純配列と間接配列とシステム配列があり，単純配列はARRAY宣言で，間接配列はVAR宣言か仮引数で間接変数として宣言してから使用する。システム配列は宣言しない。

1バイト型は1バイト単位で，2バイト型は2バイト単位で，配列要素をアクセスする。

アクセス時に添字のチェックはしない。システム配列はメモリやI/O, S-OS特殊ワークエリアを配列の形で直接アクセスする。

単純配列はARRAY宣言で宣言する。たとえば，

ARRAY BYTE BUFF[10];

と宣言すると，1バイト型の配列がBUFF[0]からBUFF[10]までの11個確保される。単純配列名は配列のワークの先頭のアドレスを指す定数として扱われるが，動的な（局所宣言で宣言された）配列名は定数式には使用できない。

間接配列は間接変数を配列として使用する。間接変数の値をインデックスとしてメモリをアクセスする。FuzzyBASICのメモリ配列と同じ。型は間接変数を宣言する際に指定する。省略した場合は2バイト型と見なされる。また，間接配列名は変数である。

ARRAY宣言

単純配列を宣言する。複数の配列を宣言する場合は，（カンマ）でつなぐ。

ARRAY BYTE ABUF[5], WORD C[3];

のように型配列名［定数式］の形で宣言すると，定数式＋1個分の配列が確保される。添字を省略すると0と見なされ，1個分の配列が確保される。型を省略すると，2バイト型と見なされる。

後ろに：定数式とすると，配列の格納アドレスを指定することができる。

ARRAY ABC[10]:$C000;

とすると，$C000以降を配列ABCのワークエリアとし，ABC[0]の格納アドレスは$C000と¥C001，ABC[1]は$C002と$C003，ABC[2]は$C004と$C005，……となる（配列ABCは2バイト型のため)。

この場合，添字は意味を持たないので

ARRAY ABC[ ]:$C000;

としてもよい。

また＝{CODEリスト｝とすると，配列を初期化することができる。ただし，｛ ｝は文括弧。この場合，WORK宣言がなされていても，その配列のワークエリアは，プログラム中に埋め込まれる。

ARRAY BYTE DT[4]={0, 1, 2, 3, 4};

初期値が足りない場合は，残りは0で埋められる。多すぎる場合は，エラーとなる。添字が省略された場合はチェックしない。

ただし，配列の格納アドレス指定と，初期化は，大域宣言と静的宣言のみで使用でき，局所宣言では使用できない。

関数

関数頭書き

定義する関数名を宣言する。ブロックの最初に書き，以後，静的宣言や局所宣言，関数定義が続く。

関数名（仮引数リスト）

の形で書く。

仮引数リストの書式は，局所宣言のVAR宣言の書式と同じ。

仮引数を持たない場合は，

関数名( )

と書く。

仮引数は関数コール時の実引数の値を持ち(値渡し)，自動的に動的な局所変数として宣言される。

MACHINE宣言されたマシン語関数は，

関数名（引数の数）

と書く。ただし，引数の数が0個の場合と，引数の数を省略して宣言する場合は，

関数名( )

と書く。

関数定義

関数を定義する。

```
BEGIN
  局所宣言;
  文;
  ：
  文;
END;
```

の形で書く。

END(式);

とすると，式の値を関数の値として返すことができる。

関数

ユーザー関数とシステム関数とMACHINE関数がある。

ユーザー関数はプログラム中で定義した関数。引数を渡すのにIYレジスタをポインタとして使用する。

システム関数には，CODE関数とPRINT関数がある。

MACHINE関数はMACHINE宣言した関数で，ユーザー関数と異なり，レジスタやスタックを使って引数を渡す。主に，外部のマシン語サブルーチンをMACHINE関数として宣言するが，プログラム中でCODE関数を使って定義したマシン語関数もMACHINE関数とすることができる。

関数コール

値渡しである。

関数名（実引数リスト）

の形で関数を呼び出す。

実引数と仮引数の数が合わないと，エラーになる。

RETURN(式);

や

END(式);

によって返される値が関数の値となる。

MACHINE関数で引数の数を省略して宣言した場合のみ，引数の数のチェックを行わない。

MACHINE宣言

マシン語関数を宣言する。複数のマシン語関数を宣言する場合は，（カンマ）でつなぐ。

関数名（引数の数）

の形で宣言する。ただし，その関数を使用，定義する前に宣言しなければならず，また，大域宣言でのみ宣言できるので，通常，アドレス宣言の次の大域宣言で宣言する。

MACHINE MSUB(2):$C000;

のように後ろに：定数式を付けると，外部にあるマシン語サブルーチンを関数として利用できる。上の場合MSUBは$C000にあり，引数を2個持つ関数となる。

引数の数が0個の場合は，マシン語関数ではなく，引数を持たないふつうの関数として扱われるので，プログラム内の関数の宣言は無意味である。外部の関数の宣言に使う。

MACHINE MON(0):$1F8E;

引数の数が1個から3個までの場合は，レジスタを使って引数を渡す。

引数の数が4個以上の場合は，スタックを使って引数を渡す。

引数の数を省略した場合は，スタックを使って引数を渡し，HLレジスタに引数の数が代入される。この場合に限り，引数の数のチェックは行わない。

MACHINE PRINTF( );

関数コールの実際

引数や，動的な変数や配列のポインタとしてIYレジスタを使用している。

実引数の場合，

SUB(A, B)

とすると，

```
LD    HL, (VARA)
LD    (IY+$70), L
LD    (IY+$71), H
LD    HL, (VARB)
LD    (IY+$72), L
LD    (IY+$73), H
CALL  SUB
```

というコードが生成される。

一方，関数側では，たとえば，

```
SUB(I, J)             (*仮引数*)
  VAR K;              (*静的宣言*)
  BEGIN
    VAR L;            (*局所宣言*)
    ｜
  END;
```

となっていたとすると，動的な変数は3個（I, J, L)となり，関数の最初と最後に，

```
PUSH IY
LD   BC, 6 ; 3個×2バイト
ADD  IY, BC
     ｜
POP  IY
```

というコードが生成される。

ただし，動的な変数や配列がない場合は，なにも生成されない。

この関数内での動的な変数のアドレスは次のようになる。

I ……	(IY+$6A)	下位
	(IY+$6B)	上位
J ……	(IY+$6C)	下位
	(IY+$6D)	上位
L ……	(IY+$6E)	下位
	(IY+$6F)	上位

つまり，Iの初期値は実引数Aの値，Jの初期値は実引数Bの値，Lの初期値は不定となる。

実引数用のワークは（IY+$70）から（IY+$7F）までであるため，引数の数は最大8個までとなる。実引数に関数を使用する場合，引数が8個以下でもワークがあふれてしまいエラーになる場合があるので注意すること。

動的な変数や配列のワークは（IY+$80）から（IY+$6F）までの240バイトしかないので，大きな動的配列を宣言する場合は注意すること。

関数の値は，関数から戻ってきたときのHLレジスタの値となる。

MACHINE関数の場合は，宣言した引数の数によって呼び出し方が異なる。

0個	……CALLのみ
1個	……HLレジスタに引数を代入してCALL
2個	……順にHL, DEに代入してCALL
3個	……順にHL, DE, BCに代入してCALL
4個以上	……スタックに積んでCALL
省略	……スタックに積み，HLに引数の数を代入してCALL

例）スタックに積んだようす　SUB(A, B, C);

アドレス大	
	A
	B
	C
	リターンアドレス
SP	
アドレス小	

MACHINE関数で，動的な変数や配列を使用する場合も，ユーザー関数と同様のIYレジスタの退避が行われるので注意すること。

関数の値は，関数から戻ってきたときのHLレジスタの値となる。

演算子

式

式はすべて16ビット長で演算を行う。

真は1，偽は0。

ビット演算子

AND	論理積
OR	論理和
XOR	排他的論理和
CPL	ビット反転

THE SENTINEL

<<　左シフト
>>　右シフト
HIGH　上位8ビットを値とする
LOW　下位8ビットを値とする

論理演算子（真のとき1，偽のとき0を値とする）
NOT　論理否定

関係演算子（真のとき1，偽のとき0を値とする）
==　等しい
<>　等しくない
!=　〃
>　大きい
>=　大きいか等しい
<　小さい
<=　小さいか等しい

代入演算子
=　代入

カンマ演算子
,　左から右へ計算され，最も右の論理項を値とする

算術演算子
+(単項)　正符号
-(単項)　負符号
+　加算
-　減算
*　乗算
/　除算
MOD　剰余算

ピリオド演算子（符号付きで演算を行う）
.*. ./. .MOD. .<<. .>>. .<=. .>=. .<. .>.

その他
++　インクリメント演算
変数や配列の値に1を加える。変数や配列の前に置いた場合は，+1してから値が参照され，後ろに置いた場合は，値が参照されてから+1する
--　デクリメント演算子
変数や配列の値から1を引く。前置き，後ろ置きの規則は++と同じ
&　アドレス演算子
変数や配列が格納されているアドレスを値とする。ただし，システム配列には使用できない
?:　C言語の条件演算子と同じ。三項演算子

演算の優先順位
1. ( ) [ ]
2. ++ -- & '
3. + - HIGH LOW NOT CPL（すべて単項演算子）
4. * / MOD<< >> .*. ./. .MOD. .<<. .>>.
5. + -
6. == < > != <= >= < > .<=. .>=. .<. .>.
7. AND OR XOR
8. ?:（三項演算子）
9. =
10. ,（カンマ）

システム構成

文
{ }は文括弧を表す。文括弧として[ ], ( ), 「 」, BEGIN END;が使用できる。[ ]は省略可を表す。

ラベル
ラベル名：
GOTO文やEXIT TO文のジャンプ先を指定する。ラベル名は局所的な名前となる。

式文
式；
式の文。

複合文
{文 [, 文, …, 文]}
複数の文を文括弧でくくり，ひとつの文として扱う。

空文
;
なにもしない文。

IF文
IF式 [THEN] 文1 [ELSE 文2] [ENDIF;]
式の値が真ならば文1，偽ならば文2を実行する。
文2がIF文の場合，ELSE IFをELSEIFまたはEFと書くことができる。

FOR文
FOR単純変数名=式1 TO　式2[DO]文[NEXT;]
〃　〃　〃 DOWNTO〃　〃 〃 〃
単純変数の値を式1から式2になるまで1ずつ増やし，文を繰り返す。DOWNTOの場合は1ずつ減らす。
まず，文を実行してから，終値の判定を行う。ただし，式1と式2が間に0をはさむ場合は，期待される繰り返しは行われず，1回で繰り返しを終了する。

WHILE文
WHILE 式 [DO] 文 [WEND;]
式の値が真のあいだ，文を繰り返す。

REPEAT文
REPEAT 文 UNTIL 式;
式の値が真になるまで，文を繰り返す。

EXIT文
EXIT;
FOR文，WHILE文，REPEAT文から脱出する。C言語のbreak文と同じ。
EXIT TO ラベル名;
ラベルにジャンプする。ただし，あと戻りはできない。

RETURN文
RETURN;
その関数を終了して，呼び出した関数に戻る。
RETURN(式);
式をその関数を値として，呼び出した関数に戻る。

GOTO文
GOTO ラベル名;
ラベルにジャンプする。EXIT TO ラベル名;と違って，ジャンプ先に制限はない。

CASE文
```
CASE 式0 [OF] {
    定数式1 [:] 文1
   [定数式2 [:] 文2]
   [  ⋮      ⋮   ⋮ ]
   [OTHERS [:] 文 ]
}
```
式0の値が定数式nと等しければ，文nを実行し，CASE文を脱出する。上から順に比較していき，いずれの定数式とも等しくなかった場合は，OTHERSの後ろの文を実行する。
定数式1　TO　定数式2[:] 文1
とすると，式0の値が定数式1以上，定数式2以下の場合，文1を実行する。
定数式1，[定数式2，…，] 定数式n[:] 文1
とすると，式0の値が定数式1から定数式nまでのいずれかに等しい場合，文1を実行する。

登録済みの名前
システム関数やシステム配列など，登録済みの名前は，すべて大域的な名前である。

登録済みの記号定数
FALSE　値は0
TRUE　値は1

システム配列
MEM[式]　式の値のアドレスの内容を1バイト単位でアクセスする
MEMW[式]　式の値のアドレスの内容を2バイト単位でアクセスする。式のアドレスが下位バイト，式+1のアドレスが上位バイトに対応する
PORT[式]　式の値のI/Oポートを1バイト単位でアクセスする
PORTW[式]　式の値のI/Oポートを2バイト単位でアクセスする。式のI/Oポートが下位バイト，式+1のI/Oポートが上位バイトに対応する。下位バイト，上位バイトの順にアクセスされる
SOS[式]　式の値のS-OS特殊ワークエリアを1バイト単位でアクセスする
SOSW[式]　式の値のS-OS特殊ワークエリアを2バイト単位でアクセスする。式の特殊ワークエリアが下位バイト，式+1の特殊ワークエリアが上位バイトに対応する

登録済みの変数
^A　CALL関数，GETREG関数で使用。CALL関数では値をAレジスタに代入してからマシン語ルーチンをコールし，終了後Aレジスタの値が代入される。^は↑でも可
^BC　^Aと同様
^DE　^Aと同様
^HL　^Aと同様
^IX　^Aと同様
^IY　^Aと同様
^AF　^Aと同様。^AFの上位バイトと^Aの下位バイトは同じ値を持つ
^SP　CALL関数，GETREG関数で使用。現在のSPの値が代入される
^CARRY, ^CY　CALL関数，GETREG関数で使用。CYフラグが立っていれば1，立っていなければ0が代入される
^ZORO　Zフラグ。^CARRYと同様
@KBUFF　キー入力用バッファのアドレスを値として持つ。代入すると，S-OSの#KBFADの値が変わってしまうので注意すること

登録済みの基本関数
BEEP( )
BEEP音を鳴らす。S-OSの#BELL
STOP( )
プログラムの実行を終了する
LOCATE（X座標，Y座標）
カーソルを移動する
INKEY(n)
入力されたキーの値を返す
n=0のときS-OSの#GETKYと同じ
n=1のときS-OSの#FLGETと同じ
その他のときS-OSの#INKEYと同じ
INPUT( )
キーボードから入力された数値を返す。先頭に$を付けると，16進数と見なす。コールした時点のカーソル以降を読み込み，正常な入力が行われた場合は^CARRY=0，ブレイクキーが押されたり誤入力があった場合は^CARRY=1となる
GETL（格納アドレス）
キーボードから1行入力し，格納アドレスに格納し，行の長さを返す
ブレイクキーが押された場合は-1を返す。行の最後は0となる
GETLIN（格納アドレス，長さ）
1行の最大長を指定できるほかは，GETL関数と同じ。オーバーした分は無視される
LINPUT（格納アドレス，長さ）
コールした時点のカーソル以降を読み込むほかはGETLIN関数と同じ
WIDTH(n)
画面モード（40キャラ，80キャラ）を切り換える。nが40以下だと，40キャラ，40より大きいと80キャラとなる。S-OSの#WIDCH
SCREEN（X座標，Y座標）
画面のキャラクタを読み出し，キャラクタコードを返す。S-OSの#SCRN
PRMODE(n)
PRINT関数の出力を切り換える
n=0のとき，画面にのみ出力

n＝1のとき，画面とプリンタに出力
その他のとき，プリンタのみに出力

BIT（値，n）
値の第nビットを調べ，0か1を返す
nの値は0から15まで

SET（値，n）
値の第nビットを1にする

RESET（値，n）
値の第nビットを0にする

ABS(n)
nを2の補数表現の符号付きの値と見なし，その絶対値を返す

SEX(n)
nを符号付き1バイトの値と見なし，符号付き2バイトの値にして返す

SGN(n)
nを符号付きの値と見なし，正なら1，0なら0，負ならー1を返す

RND(n)
0からn−1までの乱数を返す

VTOS（値，BUFF）
値を10進数の文字列に直してBUFFに格納する。文字列の最後は$00になる。BUFFは6バイト必要

GETREG( )
各レジスタなどの値，それぞれ変数^AF，^BC，^DE，^HL，^IX，^IY，^CARRY，^ZERO，^SPに代入する。単独で用いること

CALL（アドレス）
各レジスタに，変数^A，^BC，^DE，^HL，^IX，^IYの値を代入して，アドレスをコールする。コールが終了すると，GETREG( )と同様の処理をし，HLレジスタの値を返す

システム関数

CODE関数とPRINT関数がある。

CODE関数

直接データをオブジェクトに落とすための関数。式中で使われる場合は，マシン語データを実行後，HLレジスタの値を値とする。

CODEリスト

CODE関数や配列の初期化など，データを直接オブジェクトに落とすための書式。CODE項を，(カンマ)でつなぐ。

CODE項

"文字列"
文字列をそのまま，オブジェクトに落とす。文字列定数のように，自動的に最後に$00を付けることはしない

[式]
式の値をHLレジスタに代入するようなオブジェクトを作る。その他のレジスタの値は保証されない

〈ラベル名〉
ラベルのアドレスを，下位バイト，上位バイトの順で2バイトのオブジェクトにする

型，定数式
1バイト型なら，定数式の値の下位バイトを1バイトのオブジェクトにし，2バイト型なら，下位バイト，上位バイトの順で2バイトのオブジェクトにする。型を省略した場合は1バイト型と見なされる

PRINT関数

文字や数値を画面やプリンタに出力する。PRMODE関数で出力先を変えることができる。

書式リスト

PRINT関数の書式。書式項を，(カンマ)でつなぐ。

書式項

"文字列"	文字列をそのまま出力
/(スラッシュ)	改行する
値	値を10進左詰め出力
FORM$(値，n)	値を10進n桁右詰め出力
DECI$(値) %(値)	値を10進5桁右詰め出力
PN$(値)	値を符号付き10進左詰め出力
HEX2$(値)	値を16進2桁出力
HEX4$(値)	値を16進4桁出力
MSG$(値)	値のアドレスから$0Dの直前までをASCII出力
MSX$(値) ！(値)	値のアドレスから$00の直前までをASCII出力
STR$(値，n)	値のキャラクタをn個出力
CHR$(n)	値を上位バイト，下位バイトの順にASCII出力
SPC$(n)	空白をn個出力
CR$(n)	改行をn個出力
TAB$(n)	カーソルをn回右へ移動

エラーメッセージ

Missing "文字"	あるべき文字がない
Syntax error	文法エラー
Illegal constant	正しい定数式ではない
Illegal brace	文括弧エラー。あるべき文括弧がない。または開きと閉じの括弧があわない
Bad string	文字列エラー。$20以下のコードがある
Illegal name	名前を誤使用している
Dup def name	二重に宣言している
Undef array	未宣言配列
Undef var	未宣言変数
Illegal address	アドレス宣言のアドレス指定が正しくない
Too many arguments	引数が多すぎる。引数のワークは8個分しかない
Too many data	データが多すぎる
Out of range	値が大きすぎる
Local area overflow	局所域がいっぱいになった。動的局所域は240バイトしかない
Unmatched arguments	引数の数が合わない
Dev by 0	0で割っている
Missing UNTIL	UNTILがない
Missing TO/DOWNTO	TO/DOWNTOがない
Can't jump	ジャンプできない
Nesting overflow	ループの入れ子が深すぎる。16レベルまで
Global table overflow	大域表がいっぱいになった
Local table overflow	局所表がいっぱいになった
Too long line	1行が長すぎる。1行は255文字以内
Too long name	名前が長すぎる。名前は32文字以内
Can't include	INCLUDEできない。オンメモリ版ではINCLUDEできない。入れ子は8レベルまで
Undef func	未宣言関数
Undef label	未宣言ラベル
Memory over	メモリがオーバーした

THE SENTINEL

リスト1 SLANG

```
3000 C3 1A 30 C3 54 30 01 01 : 56
3008 00 B0 3B 71 00 72 00 72 : 40
3010 00 00 00 02 00 30 0A 00 : 3C
3018 01 00 11 25 00 2A 68 1F : E8
3020 B7 ED 52 22 46 63 2A 14 : FF
3028 30 22 7E 5F 22 51 63 2A : 2F
3030 12 30 B7 ED 52 22 4A 63 : 07
3038 2A 51 63 ED 5B 12 30 B7 : 1F
3040 ED 52 22 53 63 CD 90 5E : D2
3048 AF 32 43 31 CD D6 1F CD : E4
3050 E2 1F 0C 00 CD EB 1F CD : B1
3058 E2 1F 53 4C 61 6E 67 20 : F6
3060 43 6F 6D 70 69 6C 65 72 : 3B
3068 20 76 65 72 20 31 2E 30 : 1C
3070 30 0D 00 ED 73 92 30 ED : 4C
3078 7B 92 30 CD D6 1F CD EB : B7
---------------------------------
SUM: 55 A0 2C 22 99 2E 3F 7C BD0B

3080 1F 3E 5D CD F4 1F ED 5B : E2
3088 76 1F CD D3 1F CD 9B 30 : EC
3090 18 E5 00 00 3E 0D CD 33 : 48
3098 20 18 DC 1A FE 5D C0 13 : 5C
30A0 1A 13 FE 21 CA FA 1F FE : 2D
30A8 4D CA 8E 1F FE 47 CA 0C : DF
30B0 31 FE 4A CA 0C 31 FE 53 : D1
30B8 CA 44 31 FE 58 CA 16 31 : A6
30C0 FE 23 CA 27 31 F5 3A 43 : B5
30C8 31 B7 C4 D9 1F F1 FE 43 : D6
30D0 CA C3 31 FE 44 CA DE 30 : D8
30D8 FE 4F CA FA 5E C9 1A FE : 50
30E0 56 20 09 13 CD F6 30 CD : 52
30E8 27 20 18 09 CD F6 30 CD : 28
30F0 06 20 DC 96 30 C9 CD BA : 18
30F8 31 13 1A 1B FE 3A 20 05 : D6
---------------------------------
SUM: DA D8 AD 87 35 FA 8F 6C 8737

3100 1A 13 13 18 03 CD 24 20 : 6C
3108 32 5D 1F C9 CD BA 31 CD : FC
3110 B2 1F D8 C3 81 1F CD BA : 93
3118 31 CD B2 1F 38 03 22 0E : 3A
3120 30 2A 0E 30 C3 BE 1F 3A : 72
3128 43 31 2F 32 43 31 B7 20 : 20
3130 0A CD E2 1F 4F 46 46 0D : C0
3138 00 18 07 CD E2 1F 4F 4E : 8A
3140 0D 00 C9 00 21 94 30 E5 : A0
3148 3E 01 CD A3 1F 1A FE 3A : 20
3150 C0 13 CD B2 1F D8 22 70 : DB
3158 1F 22 6E 1F 22 B8 31 1A : F3
3160 FE 3A C0 13 CD B2 1F D8 : 81
3168 ED 4B 70 1F ED 42 23 22 : 3B
3170 72 1F 1A FE 3A 20 15 13 : 2B
3178 CD B2 1F D8 22 6E 1F 1A : 3F
---------------------------------
SUM: 00 28 1C 8D 57 BD A6 3A 1518

3180 FE 3A 20 08 13 CD B2 1F : 11
3188 D8 22 B8 31 CD E2 1F 57 : 08
3190 72 69 74 69 6E 67 20 00 : AD
3198 CD 9D 1F CD AF 1F DC 96 : 96
31A0 30 2A B8 31 22 70 1F CD : C1
31A8 AC 1F DC 96 30 CD E2 1F : 3B
31B0 0D 4F 4B 21 0D 00 E1 C9 : 7F
31B8 00 00 1A FE 20 20 03 13 : 6E
31C0 18 F8 C9 3E 01 32 53 58 : F5
31C8 3E 00 32 54 58 AF 32 67 : 64
31D0 59 3E 01 32 D6 5A 1A FE : 12
31D8 2F 20 06 13 3E 00 32 D6 : AE
31E0 5A 1A B7 20 07 3E 01 CD : 5E
31E8 8A 65 18 19 3A 06 30 FE : 8E
31F0 01 20 0A 3E 00 32 68 63 : 66
31F8 CD A8 59 18 08 3E 01 32 : 5F
---------------------------------
SUM: 8E 97 98 BB 32 81 1D C7 03D0

3200 68 63 CD EA 59 CD 90 5E : 96
3208 CD 7A 36 CD 30 5A 3E 00 : 12
3210 32 63 63 32 64 63 21 00 : 12
3218 00 22 55 63 22 5F 63 23 : E1
3220 22 57 63 2A 08 30 22 5B : BB
3228 63 22 4E 63 E5 FD E1 CD : C6
3230 50 33 CD FF 63 CD 5F 64 : 42
3238 CD 98 58 CD CC 33 CD 90 : E6
3240 36 CD 7A 36 CD AE 56 B7 : 3B
3248 20 F1 3A 54 58 FE 01 20 : 16
3250 18 CD F1 62 0D 4D 69 73 : 6E
3258 73 69 6E 67 20 23 45 4E : 87
3260 44 49 46 20 20 20 20 0D : 60
3268 00 2A 57 63 7C B5 28 18 : 55
3270 CD F1 62 0D 55 6E 64 65 : B9
3278 66 20 66 75 6E 63 0D 00 : 3F
---------------------------------
```

```
SUM: 61 1E 09 FD DC D8 3F BF AD53

3280 3E 00 32 A5 5D CD B1 5E : 4E
3288 CD E2 1F 0D 50 72 6F 67 : 73
3290 72 61 6D 20 00 2A 5B 63 : 48
3298 CD BE 1F CD E2 1F 20 2D : C5
32A0 20 00 FD E5 E1 2B CD BE : 99
32A8 1F ED 5B 5F 63 7A B3 28 : 7E
32B0 23 CD E2 1F 0D 4F 62 6A : 19
32B8 65 63 74 20 20 00 2A 5B : 01
32C0 63 19 CD BE 1F CD E2 1F : F4
32C8 20 2D 20 00 FD E5 E1 2B : 5B
32D0 19 CD BE 1F CD E2 1F 0D : 9E
32D8 57 6F 72 6B 20 20 20 20 : 23
32E0 00 3A 63 63 FE 01 20 0D : 2C
32E8 2A 5D 63 CD BE 1F CD E2 : 43
32F0 1F 20 2D 20 00 CD EB 35 : 79
32F8 CD BE 1F CD E2 1F 20 2D : C5
--------------------------------
SUM: 1A 15 BA 87 A7 3C A1 C8 2212

3300 20 3F 3F 3F 3F 0D 0D 43 : 79
3308 6F 6D 70 6C 65 74 65 20 : 16
3310 21 20 20 20 20 45 72 72 : CA
3318 6F 72 00 2A 55 63 CD DC : 6C
3320 5A CD EB 1F CD EB 35 11 : 2F
3328 90 FF 19 ED 5B 5B 63 FD : AB
3330 21 16 00 FD 19 CD 9C 4B : 01
3338 3A 64 63 FE 01 20 0E FD : 2B
3340 21 11 00 FD 19 3E 31 2A : E1
3348 61 63 CD 99 4B C3 C4 1F : 1B
3350 CD 6F 5C 4F 52 C7 91 33 : C4
3358 57 4F 52 CB 9B 33 4F 46 : 26
3360 46 53 45 D4 8A 33 53 54 : 16
3368 41 43 CB AA 33 00 30 08 : 64
3370 CD 81 1F CD E4 4C 18 D8 : 5A
3378 2A 5B 63 ED 5B 5F 63 19 : 0B
--------------------------------
SUM: 88 28 43 E4 A8 35 C6 16 9D1C

3380 7D D6 00 7C DE 70 DA 49 : 40
3388 61 C9 CD B6 33 22 5F 63 : C4
3390 C9 CD BF 33 22 5B 63 E5 : 4D
3398 FD E1 C9 3E 01 32 63 63 : DE
33A0 CD BF 33 22 5D 63 22 4E : 11
33A8 63 C9 3E 01 32 64 63 CD : 31
33B0 BF 33 22 61 63 C9 CD EF : 5D
33B8 38 FE 01 C2 77 30 C9 CD : 36
33C0 B6 33 7D D6 00 7C DE 30 : C6
33C8 DA 49 61 C9 3E 00 C3 D3 : 21
33D0 33 3E 01 32 A5 5D CD 6F : E2
33D8 5C 56 41 D2 0B 34 41 52 : 97
33E0 52 41 D9 62 34 43 4F 4E : E2
33E8 53 D4 70 35 00 38 14 3A : 52
33F0 A5 5D FE 01 C8 CD 6F 5C : 61
33F8 4D 41 43 48 49 4E C5 96 : 0B
--------------------------------
SUM: 81 C9 93 6C D0 82 60 09 3628

3400 35 00 D0 CD 81 1F CD E4 : 23
3408 4C 18 CB CD 39 36 FE 01 : 6A
3410 20 04 3E 02 18 0A FE 02 : 86
3418 20 04 3E 03 18 02 3E 01 : BE
3420 CD FB 35 3A A1 5D FE 01 : 34
3428 C4 C1 34 CD AE 56 FE 3A : C2
3430 20 07 DD 23 CD 20 36 18 : 62
3438 1D FE 3D 20 13 DD 23 3A : C5
3440 A1 5D FD E5 E1 CD 26 36 : EA
3448 CD EF 38 CD 9C 4B 18 06 : C6
3450 21 02 00 CD 55 35 CD 5C : A3
3458 34 38 B0 C9 CD 11 5C 2C : 4B
3460 0D C9 CD 5C 36 FE 01 20 : 54
3468 04 3E 21 18 02 3E 22 CD : AA
3470 FB 35 CD C1 34 CD A8 34 : 9B
3478 CD AE 56 FE 3A 20 07 DD : 0D
--------------------------------
SUM: 2B 51 90 64 5E 98 95 37 3E9F

3480 23 CD 20 36 18 1C FE 3D : B5
3488 20 07 DD 23 CD 21 35 18 : 62
3490 11 2A BF 34 7C B5 20 07 : 86
3498 23 22 BF 34 CD A8 34 CD : AE
34A0 55 35 CD 5C 34 38 BB C9 : A3
34A8 2A BF 34 3A A1 5D FE 21 : 74
34B0 C8 FE 23 C8 FE 31 C8 FE : A6
34B8 33 C8 29 22 BF 34 C9 00 : 02
34C0 00 CD 00 35 DD 7E 00 FE : 5B
34C8 5B C0 2A BF 34 E5 3A A1 : F8
34D0 5D C6 02 32 A1 5D 2A 9F : 1E
34D8 5D CD 26 36 CD 00 35 2A : B2
34E0 BF 34 24 25 C4 8E 61 2C : 1B
34E8 2D 20 01 2C E5 7D CD 93 : 3C
34F0 37 E1 D1 CD DD 69 7C FE : 76
34F8 80 D4 8E 61 22 BF 34 C9 : 21
--------------------------------
SUM: A9 03 9E 1C E7 87 48 FF A7E2

3500 3E 5B CD 83 56 CD AE 56 : 10
3508 21 FF FF FE 5D 28 09 CD : 78
3510 EF 38 7C FE 80 D4 8E 61 : E4
3518 23 22 BF 34 3E 5D C3 83 : 19
3520 56 3A A1 5D FD E5 E1 CD : 1E
3528 26 36 CD 86 4C F5 CD 81 : 3E
3530 65 2A BF 34 7C B5 28 15 : F0
3538 B7 ED 42 30 05 CD 78 61 : C1
3540 18 0B 28 09 AF CD A1 4B : BC
3548 2B 7C B5 20 F7 CD BA 4C : 46
3550 F1 CD 68 4C C9 3A 63 63 : 3B
3558 FE 01 20 0A ED 5B 4E 63 : 22
3560 19 22 4E 63 18 09 AF CD : 89
3568 A1 4B 2B 7C B5 20 F7 C9 : 28
3570 3E 41 21 00 00 CD 00 36 : A3
3578 3E 3D CD 83 56 CD 70 4C : AA
--------------------------------
SUM: 71 7B 42 DB BA 74 78 40 E903

3580 30 0B 21 00 00 22 BF 34 : 71
3588 CD 21 35 18 03 CD 20 36 : 61
3590 CD 5C 34 38 DB C9 AF 32 : 1A
3598 A2 5D 3E C1 21 00 00 CD : EC
35A0 00 36 3E 28 CD 83 56 CD : 0F
35A8 AE 56 FE 29 28 23 CD EF : 32
35B0 38 7C B5 20 0A 3E C0 21 : B2
35B8 00 00 CD 26 36 18 12 7C : CF
35C0 FE 00 20 03 7D FE 10 20 : CC
35C8 01 37 D4 60 61 7D CD 93 : AA
35D0 37 3E 29 CD 83 56 CD 11 : 22
35D8 5C 3A 0D 30 08 3E 81 32 : CC
35E0 A1 5D CD 20 36 CD 5C 34 : 7E
35E8 38 AC C9 3A 63 63 FE 01 : AC
35F0 20 05 2A 4E 63 18 03 FD : 18
35F8 E5 E1 C9 F5 CD EB 35 F1 : 62
--------------------------------
SUM: C2 8B 39 A5 66 F6 40 DB 18A1

3600 32 A1 5D 22 9F 5D CD 80 : 9B
3608 5F CD B3 5D 3A A1 5D 2A : 9E
3610 9F 5D 38 05 CD E1 5C 18 : 5B
3618 06 CD 26 36 CD 37 61 C9 : 5D
3620 CD EF 38 3A A1 5D EB 2A : 41
3628 A3 5D 2B 2B CD 9A 1F 2B : 07
3630 7A CD 9A 1F 2B 7B C3 9A : 03
3638 1F CD 5C 36 32 A1 5D CD : 7B
3640 80 5F FE 01 20 0A 3A A1 : E3
3648 5D B7 C4 D3 60 AF 18 08 : DA
3650 3A A1 5D B7 20 02 3E 02 : 51
3658 32 A1 5D C9 CD 6F 5C 42 : D3
3660 59 54 C5 01 00 57 4F 52 : 6B
3668 C4 02 00 21 0D 01 00 25 : 1A
3670 0D 02 00 00 38 02 2E 00 : 77
3678 7D C9 21 00 00 22 59 63 : 45
--------------------------------
SUM: 2F F7 29 EA F0 CF D3 0E 4041

3680 2A 10 30 22 4C 63 AF 32 : 1C
3688 50 63 32 21 52 C3 9A 1F : D4
3690 CD 7A 36 CD 00 37 CD D1 : 1F
3698 33 CD 86 4C F5 CD E3 37 : AE
36A0 CD AD 37 3A 50 63 CD 4C : B7
36A8 44 3E 01 CD 11 51 CD BA : 39
36B0 4C 38 05 CD FA 4B 18 F6 : A9
36B8 F1 32 FF 36 CD 68 4C CD : A6
36C0 02 51 CD AE 56 FE 28 20 : 6A
36C8 0B CD D5 38 CD 74 39 3E : 9D
36D0 02 32 FF 36 CD F8 4C 3A : B4
36D8 FF 36 FE 02 CC E4 4C 2A : 5B
36E0 59 63 7C B5 28 18 CD F1 : EB
36E8 62 55 6E 64 65 66 20 6C : E0
36F0 61 62 65 6C 0D 00 3E 01 : E0
36F8 32 A5 5D CD B1 5E C9 00 : D9
--------------------------------
SUM: 24 54 A5 D6 C2 BB E4 42 82ED

3700 3E 00 32 5C 37 CD 80 5F : AF
3708 CD A6 5D F5 F5 C5 B7 20 : 56
3710 0A 3E C0 21 00 00 CD E1 : D7
3718 5C 18 18 FE C0 30 14 FE : 8C
3720 80 38 05 CD 37 61 18 03 : 3D
3728 CD 25 61 3E C0 21 00 00 : 72
3730 CD 26 36 3E 28 CD 83 56 : 35
3738 C1 F1 FE C1 20 05 CD 78 : DB
3740 37 18 03 CD 5D 37 F1 FE : A2
3748 C0 D4 A1 37 CD 92 37 2A : 2C
3750 A3 5D 22 5A 37 3E 29 C3 : DD
3758 83 56 00 00 00 2A A3 5D : 03
3760 E5 CD AE 56 06 00 FE 29 : E3
3768 28 09 CD 08 38 3A 50 63 : 2B
3770 CB 3F 47 E1 22 A3 5D C9 : 1D
3778 78 FE 03 20 05 3E 01 32 : 0F
--------------------------------
SUM: B9 22 8C 37 F1 62 20 FE A8F4

3780 5C 37 CD AE 56 FE 29 20 : AB
3788 04 06 00 18 04 CD EF 38 : 1A
3790 45 C9 78 2A A3 5D 2B C3 : 9E
3798 9A 1F 2A A3 5D 2B C3 94 : 65
37A0 1F CD 9A 37 B8 28 05 C5 : 67
37A8 CD C4 61 C1 C9 2A 5A 37 : 37
37B0 22 A3 5D 2B 2B CD 94 1F : F8
37B8 47 2B CD 94 1F 57 2B CD : 41
37C0 94 1F 5F D5 78 FD E5 E1 : 22
37C8 C6 C0 CD 26 36 E1 FD E5 : 72
37D0 C1 7C B5 28 0D CD D7 4B : 16
37D8 2A 57 63 2B 22 57 63 EB : D6
37E0 18 EF C9 3E 01 32 A5 5D : 43
37E8 CD 6F 5C 56 41 D2 08 38 : 41
37F0 41 52 52 41 D9 4B 38 43 : C5
37F8 4F 4E 53 D4 70 35 00 D0 : 39
--------------------------------
SUM: 4E 34 A2 41 8D 4F 25 3B 5E7E

3800 CD 81 1F CD E4 4C 18 E0 : 62
3808 3E 01 32 A5 5D CD 39 36 : AF
3810 FE 01 20 04 3E 12 18 0A : 95
3818 FE 02 20 04 3E 13 18 02 : 8F
3820 3E 11 2A 50 63 26 00 CD : 1F
3828 00 36 3A A1 5D FE 11 C4 : 41
3830 7B 38 21 02 00 CD 3E 38 : 19
3838 CD 5C 34 38 D0 C9 3A 50 : B8
3840 63 85 FE F0 D4 A7 61 32 : E4
3848 50 63 C9 CD 5C 36 FE 01 : DA
3850 20 04 3E 31 18 02 3E 32 : 1D
3858 32 A1 5D 2A 50 63 26 00 : 33
3860 CD 00 36 CD 7B 38 CD A8 : F8
3868 34 7D D6 F0 7C DE 00 D4 : A5
3870 8E 61 CD 3E 38 CD 5C 34 : 8F
3878 38 D1 C9 CD B3 38 DD 7E : E5
--------------------------------
SUM: 59 9C 4E 85 C7 55 D3 CE 0CB6

3880 00 FE 5B C0 2A BF 34 E5 : 1B
3888 3A A1 5D C6 02 32 A1 5D : 30
3890 2A 9F 5D CD 26 36 CD B3 : CF
3898 38 2A BF 34 E5 7D CD 93 : 17
38A0 37 E1 D1 CD DD 69 7D D6 : 4F
38A8 F0 7C DE 00 D4 8E 61 22 : 2F
38B0 BF 34 C9 3E 5B CD 83 56 : FB
38B8 CD AE 56 21 00 00 FE 5D : 4D
38C0 C4 EF 38 23 7D D6 F0 7C : CD
38C8 DE 00 D4 8E 61 22 BF 34 : B6
38D0 3E 5D C3 83 56 21 69 63 : 24
38D8 22 EE 46 21 9B 63 22 F0 : 87
38E0 46 AF 32 15 44 AF 32 91 : F2
38E8 4B 3E 01 32 67 63 C9 CD : 1C
38F0 D5 38 3A A1 5D 67 3A 67 : 4D
38F8 63 6F E5 2A 9F 5D E5 2A : EC
--------------------------------
SUM: 1A 75 09 1A B9 BA 22 25 94C4

3900 A3 5D E5 CD 75 65 E1 22 : 8F
3908 A3 5D E1 22 9F 5D E1 7C : 5C
3910 32 A1 5D 7D 32 67 63 CD : 76
3918 C8 46 F5 FE 01 C4 E7 60 : 0D
3920 F1 C9 CD 28 39 C3 E4 4C : DB
3928 CD D5 38 CD 74 39 3A 91 : 1F
3930 4B B7 28 3F FE EC 20 04 : 77
3938 3E 02 18 32 FE ED 20 04 : 99
3940 3E 04 18 2A FE 7C 20 04 : 22
3948 3E 03 18 22 FE 7A 20 04 : 17
```

▶11月のある日のこと，PCエンジンをやっていた子供が友達にこういった。「ゲーセンにこれとソックリなゲームがあるんだよね」と。ソフトを見ると「ビックリマンワールド」だった。おいおい……。「ワンダーボーイ」を「○○名人の冒険島」の盗作だと思っている子供もいた。う～ん，こわい。

鈴木　崇(16)東京都

```
3950 3E 03 18 1A FE 7B 20 04 : 10
3958 3E 03 18 12 FE 7E 20 04 : 0B
3960 3E 03 18 0A FE 03 20 04 : 88
3968 3E 05 18 02 3E 01 FD 2B : C4
3970 3D 20 FB C9 CD 7D 39 C3 : 67
3978 7F 49 CD E9 38 CD 9D 39 : 59
---------------------------------
SUM: B7 76 B5 06 29 FF DD EB 23FA

3980 CD 5C 34 D0 CD 7F 49 CD : 8F
3988 2E 39 CD 9A 39 CD B5 45 : CE
3990 CD 5C 34 38 F2 AF 32 49 : B1
3998 55 C9 CD E9 38 CD D6 39 : E8
39A0 CD 11 5C 3F 0D D0 CD 04 : 27
39A8 55 CD 4B 55 FD E5 CD D3 : 44
39B0 39 CD B5 45 CD 92 4B FD : A7
39B8 E5 3E 3A CD 83 56 C1 D1 : 95
39C0 C5 CD 60 4D CD D3 39 CD : E5
39C8 B5 45 D1 CD 60 4D AF 32 : 26
39D0 49 55 C9 CD E9 38 CD 16 : 38
39D8 3A CD 6F 5C 41 4E C4 01 : 26
39E0 00 4F D2 02 00 58 4F D2 : 9C
39E8 03 00 00 D0 7D FE 01 20 : 6F
39F0 07 21 6A 01 3E 08 18 0F : 00
39F8 FE 02 20 07 21 63 01 3E : EA
---------------------------------
SUM: 62 49 5D 4E BD CC 8E 8E C192

3A00 09 18 04 21 71 01 AF F5 : 5C
3A08 E5 CD 16 3A E1 CD 18 47 : 0F
3A10 F1 32 49 55 18 C3 CD 14 : 7D
3A18 3B CD 6F 5C 3D 3D 0D F5 : 4F
3A20 00 3C 3E 0D FD 00 21 3D : E2
3A28 0D FD 00 00 30 0A E5 CD : F6
3A30 14 3B E1 CD 73 3A 18 39 : FB
3A38 CD 6F 5C 3C 3D 0D 0E 01 : 2D
3A40 3E 3D 0D 0F 01 3C 0D 05 : E6
3A48 01 3E 0D 04 01 2E 3C 3D : F8
3A50 2E 0D 2D 01 2E 3E 3D 2E : 40
3A58 0D 2E 01 2E 3C 2E 0D 19 : FA
3A60 01 2E 3E 2E 0D 18 01 00 : C1
3A68 D0 E5 CD 14 3B E1 CD AC : 2B
3A70 3A 18 A6 E5 7C FE 00 20 : 77
3A78 03 7D FE F5 20 04 3E C2 : 97
---------------------------------
SUM: 90 25 44 80 D4 F0 6C A0 B1B4

3A80 18 02 3E CA 32 49 55 CD : BF
3A88 CB 49 CD 5B 47 CD 94 3A : 1E
3A90 E1 C3 F2 46 CD 79 46 FE : 66
3A98 03 20 10 3A 49 55 FE C2 : CB
3AA0 20 04 3E 05 18 02 3E 04 : C3
3AA8 32 49 55 C9 E5 CD 6D 49 : 01
3AB0 E1 38 22 7C FE 01 20 03 : D9
3AB8 7D FE 04 20 08 CD D1 46 : 8B
3AC0 21 05 01 18 10 7C FE 01 : CA
3AC8 20 03 7D FE 0E 20 06 CD : 9F
3AD0 D1 46 21 0F 01 7C FE 01 : C3
3AD8 20 03 7D FE 05 20 04 3E : 05
3AE0 D2 18 2B 7C FE 01 20 03 : B3
3AE8 7D FE 0F 20 04 3E DA 18 : DE
3AF0 1D 7C FE 01 20 03 7D FE : 36
3AF8 04 20 04 3E 07 18 0F 7C : 10
---------------------------------
SUM: 19 B4 1E 0D DF 13 55 FF CFD5

3B00 FE 01 20 03 7D FE 0E 20 : CB
3B08 04 3E 06 18 01 AF 32 49 : 8B
3B10 55 C3 1E 47 CD 41 3B CD : 93
3B18 AE 56 FE 2B 20 0E DD 23 : 5B
3B20 CD 41 3B AF 32 49 55 CD : 95
3B28 52 47 18 13 FE 2D 20 0E : 1D
3B30 DD 23 CD 41 3B AF 32 49 : 73
3B38 55 CD 5B 47 18 01 C9 18 : BE
3B40 D6 CD AA 3B CD 6F 5C 4D : 6D
3B48 4F C4 D7 00 3C 3C 0D 42 : B1
3B50 01 3E 3E 0D 4B 01 2E 2F : 33
3B58 2E 0D BF 00 2E 4D 4F 44 : 08
3B60 2E 0D DC 00 2E 3C 3C 2E : EB
3B68 0D 42 01 2E 3E 3E 2E 0D : 35
3B70 57 01 00 30 0A E5 CD AA : EE
3B78 3B E1 CD 1E 47 18 25 CD : 58
---------------------------------
SUM: 77 DD E5 9B 2D 92 0A 49 9BD0

3B80 6F 5C 2F 0D 01 00 2A 0D : 3F
3B88 00 00 2E 2A 2E 0D 00 00 : 93
3B90 00 D0 2C 2D 20 08 CD AA : C8
3B98 3B CD 64 47 18 06 CD AA : 48
3BA0 3B CD 6D 47 AF 32 49 55 : 3B
3BA8 18 9A CD 6F 5C 48 49 47 : 22
3BB0 C8 19 3C 4C 4F D7 49 3C : 14
3BB8 00 30 0C E5 CD E1 3C AF : BA
3BC0 32 49 55 CD 79 46 C9 CD : F2
3BC8 6F 5C 4E 4F 54 0D F5 00 : BE
3BD0 43 50 4C 0D EE 00 00 30 : 0A
3BD8 10 E5 CD E1 3C 3E C2 32 : 11
3BE0 49 55 CD 94 3A E1 C3 F2 : CF
3BE8 46 DD 7E 00 FE 2D 20 16 : 02
3BF0 DD 7E 01 FE 2D 28 0F DD : 9B
3BF8 23 CD E1 3C AF 32 49 55 : 8C
---------------------------------
SUM: 48 00 58 6A 99 46 96 51 B00D

3C00 21 ED 00 C3 F2 46 DD 7E : 64
3C08 00 FE 2B 20 09 DD 7E 01 : AE
3C10 FE 2B 28 02 DD 23 C3 E1 : F7
3C18 3C FE 01 20 09 CD BC 46 : 33
3C20 6C 26 00 C3 A7 46 FE 03 : 43
3C28 20 0A CD C8 46 CD E2 4B : FF
3C30 3C C3 AD 3C CD 7F 49 3A : B7
3C38 91 4B FE EB 38 07 FD 2B : 2C
3C40 CD 91 3C 18 03 CD 83 3C : 41
3C48 C9 FE 01 20 08 CD BC 46 : BF
3C50 26 00 C3 A7 46 FE 03 20 : F7
3C58 09 CD C8 46 CD E2 4B C3 : A1
3C60 AD 3C CD 7F 49 3A 91 4B : 94
3C68 FE EB 38 07 FD 2B CD 9F : BC
3C70 3C 18 03 CD 77 3C C9 CD : 6D
3C78 63 4B 02 26 00 3E 7D 32 : C3
---------------------------------
SUM: C3 38 9E 55 AE 05 31 A7 56B1

3C80 91 4B C9 CD 63 4B 01 6C : 8D
3C88 CD 77 3C 3E 7C 32 91 4B : 48
3C90 C9 CD 63 4B 01 6A CD 77 : F3
3C98 3C 3E 7A 32 91 4B C9 CD : 98
3CA0 63 4B 01 6B CD 77 3C 3E : D8
3CA8 7B 32 91 4B C9 F5 CD 63 : 77
3CB0 4B 02 FD 6E F1 CD A1 4B : 62
3CB8 CD 77 3C 3E 03 32 91 4B : CF
3CC0 C3 A9 49 CD 63 4B 01 6E : 9F
3CC8 CD 77 3C 3E 7E 32 91 4B : 4A
3CD0 C9 CD 63 4B 03 5E 23 56 : 1E
3CD8 CD 70 4B 3E ED 32 91 4B : C1
3CE0 C9 CD 11 5C 28 0D 30 08 : 70
3CE8 CD 7D 39 3E 29 C3 83 56 : 86
3CF0 CD FF 3C CD 79 46 FE 00 : 92
3CF8 20 04 AF 32 49 55 C9 CD : 39
---------------------------------
SUM: 02 6D 15 17 DF 15 23 B7 D3BA

3D00 6F 5C 22 0D 34 3D 26 0D : 9E
3D08 A4 3D 2B 2B 0D 5A 3E 2D : 09
3D10 2D 0D 62 3E 00 DA 81 1F : 54
3D18 CD 56 3D DA A0 46 CD 80 : 6D
3D20 5F FE 01 CA 53 40 FE 03 : BC
3D28 CA 30 41 CD 42 4B CD C3 : 25
3D30 43 C3 A9 49 CD 42 4B FD : 4F
3D38 E5 E1 11 06 00 19 3E 21 : 55
3D40 CD 99 4B CD 63 4B 03 C3 : F2
3D48 00 00 FD E5 CD 5E 54 D1 : 32
3D50 CD 60 4D C3 A9 49 DD 7E : 8A
3D58 00 FE 27 20 1C DD 23 21 : 82
3D60 00 00 CD 8F 3D DD 7E 00 : F4
3D68 FE 27 28 04 65 CD 8F 3D : 4F
3D70 E5 3E 27 CD 88 56 E1 37 : 0D
3D78 C9 FE 24 20 07 DD 23 CD : DF
---------------------------------
SUM: A4 28 E4 4B 69 49 6E 31 7B79

3D80 16 60 37 C9 CD BD 5F 30 : 8F
3D88 05 CD 4A 60 37 C9 C9 DD : 22
3D90 7E 00 FE 20 30 05 CD 13 : B1
3D98 61 18 07 DD 23 FE 5C CC : A6
3DA0 AE 54 6F C9 CD 80 5F FE : E4
3DA8 02 CA 01 3E FE 03 CA E5 : BB
3DB0 3D CD AE 5D D4 A6 5D B7 : A3
3DB8 CA 31 3E FE 43 CA 31 3E : B3
3DC0 FE 42 CA A0 46 FE 10 DA : D8
3DC8 A0 46 FE 20 DA 43 3E 47 : A6
3DD0 3A 19 30 FE 00 20 0B 78 : 24
3DD8 FE 30 DA A0 46 FE 40 DA : 06
3DE0 43 3E C3 D3 60 CD AE 5D : 4F
3DE8 D4 A6 5D D4 15 41 CD 4A : 18
3DF0 41 3A 19 30 FE 01 20 08 : EB
3DF8 3A 44 42 FE 00 CC D3 60 : BD
---------------------------------
SUM: 19 94 2F BB 12 B6 0F 46 E99F

3E00 C9 CD A6 5D FE 80 20 05 : 3C
3E08 CD A0 46 18 1A FE 81 20 : 84
3E10 05 CD A0 46 18 11 CD F8 : A6
3E18 43 CD A9 49 CD 42 4B 3E : 9A
3E20 21 CD A1 4B CD 1C 44 3E : 45
3E28 28 CD 83 56 3E 29 C3 83 : 7B
3E30 56 E5 F5 CD 42 4B 3E 21 : E9
3E38 CD A1 4B F1 E1 CD 33 52 : DD
3E40 C3 A9 49 CD EB 4B CD A0 : 25
3E48 46 CD 42 4B CD 63 4B 03 : 1E
3E50 FD E5 E1 CD A9 49 C3 52 : 97
3E58 47 00 3E 23 32 59 3E C3 : 34
3E60 67 3E 3E 2B 32 59 3E CD : A4
3E68 80 5F FE 03 CA 99 3E CD : 4E
3E70 DC 40 CD A2 46 CD C8 46 : AC
3E78 F5 E5 CD A2 46 CD 7F 49 : 24
---------------------------------
SUM: 4F 44 19 DD 46 0A 0D 70 6DB8

3E80 3A 59 3E CD A1 4B E1 F1 : 5C
3E88 C3 A3 40 32 59 3E CD 75 : B1
3E90 3E 3E 02 32 D4 42 C3 F9 : 82
3E98 3E CD AE 5D D4 A6 5D D4 : C1
3EA0 15 41 FE 45 CA 4A 3F CD : B9
3EA8 4A 41 3A 44 42 FE 00 CC : 15
3EB0 D3 60 CD 7F 49 3A D4 42 : 18
3EB8 FE 01 20 15 3A 59 3E FE : 03
3EC0 23 20 04 3E 34 18 02 3E : 11
3EC8 35 CD A1 4B CD C3 3C 18 : D2
3ED0 21 CD 63 4B 03 5E 23 56 : 76
3ED8 3A 59 3E FE 23 20 04 3E : 54
3EE0 13 18 02 3E 1B CD A1 4B : 3F
3EE8 CD 63 4B 03 72 2B 73 CD : 5B
3EF0 70 4B C9 32 59 3E CD B2 : CC
3EF8 3E 3A D4 42 FE 01 20 0F : BC
---------------------------------
SUM: EA FD 83 32 3C DC 85 CF 4BD4

3F00 3A 59 3E FE 23 20 04 3E : 54
3F08 2D 18 02 3E 2C 18 0D 3A : 10
3F10 59 3E FE 23 20 04 3E 2B : 45
3F18 18 02 3E 23 CD 30 3F 3A : F1
3F20 91 4B FE EB 20 04 3E EC : 13
3F28 18 02 3E 01 32 91 4B C9 : 30
3F30 08 3A 91 4B FE EB 38 0D : 4C
3F38 FD 2B 08 D6 10 CD A1 4B : CF
3F40 CD 70 4B 18 04 08 CD A1 : 1A
3F48 4B C9 7C FE 42 20 03 7D : 70
3F50 FE 4A 20 07 3E 01 21 B2 : 81
3F58 3E 18 58 7C FE 42 20 03 : 8D
3F60 7D FE 45 20 07 3E 02 21 : 48
3F68 B2 3E 18 47 7C FE 42 20 : 2B
3F70 03 7D FE D6 20 07 3E 01 : BA
3F78 21 BE 3F 18 36 7C FE 42 : 28
---------------------------------
SUM: 2D 75 2A 7D F7 E3 81 41 8106

3F80 20 03 7D FE DB 20 07 3E : DE
3F88 02 21 BE 3F 18 25 7C FE : D7
3F90 43 20 03 7D FE 32 20 07 : 3A
3F98 3E 01 21 11 40 18 14 7C : 59
3FA0 FE 43 20 03 7D FE 37 20 : 36
3FA8 07 3E 02 21 11 40 18 03 : D4
3FB0 CD D3 60 32 D4 42 E5 CD : FA
3FB8 8B 42 E1 C3 81 1F CD 7F : 5D
3FC0 49 CD 63 4B 02 44 4D 3A : 91
3FC8 D4 42 FE 01 20 21 CD 63 : 86
3FD0 4B 02 ED 68 3A 59 3E FE : 71
3FD8 23 20 04 3E 2C 18 02 3E : 09
3FE0 2D CD A1 4B CD 63 4B 02 : 63
3FE8 ED 69 CD 77 3C 18 18 CD : D3
3FF0 63 4B 05 ED 68 03 ED 60 : 58
3FF8 3A 59 3E CD A1 4B CD 63 : BA
---------------------------------
SUM: 42 E6 C5 52 AE CD 2F 99 8148

4000 4B 05 ED 61 0B ED 69 C9 : C8
4008 32 59 3E CD BE 3F C3 F9 : 4F
```

▶交通安全もかねて(?)スクーターのキーに「その筋キーホルダー」をリンクしていたのだが，ジーンズのポケットとゆーのは環境が劣悪らしく，「Oh！」の文字が削れてしまったのだ。こいつぁいけねーや，と思ったが肝心の「その筋」の部分は健在であったので，まぁよしとしよう。

田中　真一(18)三重県

```
4010 3E CD 7F 49 3A D4 42 FE : 21
4018 01 20 18 21 2C 00 CD 10 : 63
4020 47 3A 59 3E FE 23 20 04 : 5D
4028 3E 1C 18 02 3E 1D 21 3E : 2E
4030 00 18 0E 21 33 00 CD 10 : 57
4038 47 3A 59 3E D6 10 21 42 : 61
4040 00 CD A1 4B CD 10 47 C3 : A0
4048 70 4B 32 59 3E CD 11 40 : A2
4050 C3 F9 3E CD AE 5D D4 A6 : 4C
4058 5D D4 FB 40 FE 41 CA A0 : 15
4060 46 FE 21 CA A0 46 FE 22 : 35
4068 CA A0 46 FE 23 CA A0 46 : 81
4070 FE 24 CA A0 46 FE 31 CA : CB
4078 43 3E FE 32 CA 43 3E FE : FA
---------------------------------
SUM: 69 D8 D5 82 FE 1C 6D DD 9487

4080 33 CA 43 3E FE 34 CA 43 : BD
4088 3E CD E5 40 CD A2 46 CD : B2
4090 9D 43 D0 B7 C2 8B 3E CD : BF
4098 9D 39 CD 7F 49 CD D1 46 : 4F
40A0 CD C8 46 FE 02 20 07 3E : 40
40A8 22 CD 99 4B 18 1A 3A 91 : D0
40B0 4B FE EB 38 0D FD 2B 11 : B2
40B8 FD 73 CD C9 40 CD 70 4B : CE
40C0 18 06 11 FD 75 CD C9 40 : 77
40C8 C9 CD E2 4B F5 EB CD 9C : 0C
40D0 4B 25 F1 F5 CD 99 4B F1 : F8
40D8 3C C3 A1 4B CD AE 5D D4 : 97
40E0 A6 5D D4 FB 40 FE 10 30 : 50
40E8 04 3E 02 18 0D FE 20 30 : B7
40F0 04 3E 03 18 05 CD 25 61 : B5
40F8 3E 02 C9 CD F1 62 55 6E : EC
---------------------------------
SUM: 36 AF 83 7E 84 5C E3 1E 7282

4100 64 65 66 20 76 61 72 20 : B8
4108 0D 00 3E 03 21 00 00 CD : 3C
4110 00 36 3E 03 C9 CD F1 62 : 60
4118 55 6E 64 65 66 20 61 72 : E5
4120 72 61 79 0D 00 3E 03 21 : BB
4128 00 00 CD 00 36 3E 03 C9 : 0D
4130 CD AE 5D D4 A6 5D D4 15 : 98
4138 41 FE 45 CA 81 1F CD 4A : 05
4140 41 3A 44 42 FE 01 CC 52 : 1E
4148 42 C9 CD B5 41 78 32 D4 : 4C
4150 42 79 32 43 42 7B 32 D5 : F4
4158 42 C5 7B FE 01 CC EB 4B : 83
4160 7A CD A2 46 C1 79 FE 02 : 69
4168 20 06 CD 9A 37 32 A2 5D : F5
4170 CD 8B 42 3E 01 32 44 42 : 91
4178 3A 43 42 FE 02 20 1D 26 : 22
---------------------------------
SUM: EE F8 DF 8A A0 03 87 17 5CBD

4180 00 3A A2 5D 6F CD 3D 42 : F4
4188 DD 7E 00 FE 5B 20 08 CD : A9
4190 8B 42 CD 52 47 18 05 3E : 8E
4198 00 32 44 42 CD 31 42 CD : C5
41A0 52 47 3A D5 42 FE 01 20 : 09
41A8 0B CD 7F 49 CD 63 4B 04 : 1F
41B0 FD E5 D1 19 C9 16 01 1E : CA
41B8 00 06 01 0E 01 FE 21 C8 : FD
41C0 FE 31 20 03 1E 01 C9 FE : 38
41C8 02 20 03 16 02 C9 FE 12 : 16
41D0 20 03 16 03 C9 0E 02 FE : 13
41D8 23 C8 FE 33 20 03 1E 01 : 5E
41E0 C9 FE 04 20 03 16 02 C9 : CF
41E8 FE 14 20 03 16 03 C9 06 : 1D
41F0 02 0E 01 FE 22 C8 FE 32 : 29
41F8 20 03 1E 01 C9 FE 03 20 : 2C
---------------------------------
SUM: EE 6A B8 A5 C4 65 AD 54 9C12

4200 03 16 02 C9 FE 13 20 03 : 18
4208 16 03 C9 0E 02 FE 24 C8 : DC
4210 FE 34 20 03 1E 01 C9 FE : 3B
4218 05 20 03 16 02 C9 FE 15 : 1C
4220 20 03 16 03 C9 CD 25 61 : 58
4228 06 02 0E 01 16 01 1E 00 : 4C
4230 C9 21 02 00 3A D4 42 FE : 3A
4238 02 CC 3D 42 C9 CD A0 46 : C9
4240 C3 64 47 00 00 3E 02 C3 : 71
4248 4C 42 3E 01 32 D4 42 CD : E2
4250 8B 42 CD 9D 43 38 13 CD : 92
4258 7F 49 3A D4 42 FE 01 20 : 37
4260 05 CD C3 3C 18 03 CD D1 : 8A
4268 3C C9 B7 C2 F3 3E 3A D4 : BD
4270 42 F5 CD 9D 39 CD E4 49 : D4
4278 CD 63 4B 01 73 F1 FE 02 : E0
---------------------------------
SUM: 76 7E 6F 44 70 91 71 F0 F7CA

4280 20 06 CD 63 4B 02 23 72 : 38
4288 C3 70 4B 3A D4 42 47 3A : 4F
4290 D5 42 4F C5 3A 43 42 57 : 41
4298 3A A2 5D 5F D5 3A 59 3E : 3E
42A0 F5 3E 5B CD 83 56 DD 7E : 8F
42A8 00 FE 5D 20 08 21 00 00 : A4
42B0 CD A0 46 18 03 CD 7D 39 : 51
42B8 3E 5D CD 83 56 F1 32 59 : BD
42C0 3E D1 7A 32 43 42 7B 32 : ED
42C8 A2 5D C1 78 32 D4 42 79 : F9
42D0 32 D5 42 C9 00 00 3E 01 : 51
42D8 C3 DD 42 3E 02 32 D4 42 : 6A
42E0 CD 8B 42 CD 9D 43 38 25 : A4
42E8 CD 7F 49 CD 63 4B 02 44 : 56
42F0 4D 3A D4 42 FE 01 20 0B : C7
42F8 CD 63 4B 02 ED 68 CD 77 : 16
---------------------------------
SUM: 7B 1A F8 D8 74 35 87 2A B01F

4300 3C 18 09 CD 63 4B 05 ED : CA
4308 68 03 ED 60 C9 B7 C2 08 : 02
4310 40 3A D4 42 F5 CD 9D 39 : 28
4318 CD E4 49 CD 63 4B 04 44 : BD
4320 4D ED 59 F1 FE 02 20 07 : AB
4328 CD 63 4B 03 03 ED 51 C3 : 82
4330 70 4B 3E 01 C3 39 43 3E : 77
4338 02 32 D4 42 CD 8B 42 CD : B1
4340 9D 43 38 15 CD 7F 49 3A : FC
4348 D4 42 FE 01 20 05 21 2C : 87
4350 00 18 03 21 33 00 C3 44 : 76
4358 40 B7 C2 4A 40 3A D4 42 : 93
4360 FE 01 20 05 21 3E 00 18 : 9B
4368 03 21 42 00 E5 CD 9D 39 : EE
4370 CD E4 49 E1 C3 44 40 3E : 60
4378 01 C3 7E 43 3E 02 F5 3E : F8
---------------------------------
SUM: BD 23 ED 1D 7C DC 31 00 7F57

4380 5B CD 83 56 CD F2 38 F1 : E9
4388 FE 01 20 05 6E 26 00 18 : D0
4390 04 7E 23 66 6F CD A0 46 : 2D
4398 3E 5D C3 83 56 CD 6F 5C : CF
43A0 2B 2B 0D 23 00 2D 2D 0D : ED
43A8 2B 00 00 7D D8 DD 7E 00 : DB
43B0 FE 3D 20 0C DD 7E 01 FE : C1
43B8 3D 28 05 DD 23 AF 37 C9 : 19
43C0 AF B7 C9 CD A6 5D FE 44 : 41
43C8 CA 81 1F FE 81 CA 64 45 : 5C
43D0 FE C1 CA 64 45 F5 CD F8 : EC
43D8 43 CD A9 44 F1 FE 80 30 : 9C
43E0 05 CD 92 37 18 03 CD A1 : 24
43E8 37 3A 16 44 CD 4C 44 CD : F5
43F0 17 44 3A 16 44 C3 9F 44 : 95
43F8 B7 20 0A 3E C0 21 00 00 : 00
---------------------------------
SUM: F0 6A 02 0F 1E 36 89 E2 212D

4400 CD E1 5C 18 0F FE 80 30 : DF
4408 0B CD 25 61 3E C0 21 00 : 7D
4410 00 CD 26 36 C9 00 00 3E : 30
4418 CD CD A0 65 2A A3 5D 2B : F4
4420 2B CD 94 1F 47 2B CD 94 : 7E
4428 1F 57 2B CD 94 1F 5F D5 : 55
4430 78 FE C0 38 13 FD E5 D1 : 34
4438 7B CD 9A 1F 23 7A CD 9A : 05
4440 1F 2A 57 63 23 22 57 63 : 02
4448 E1 C3 9C 4B B7 28 4A F5 : A9
4450 CD 63 4B 02 FD E5 F1 FE : 4E
4458 01 20 08 CD 63 4B 02 FD : A3
4460 23 18 36 FE 02 20 0A CD : 68
4468 63 4B 04 FD 23 FD 23 18 : 0A
4470 28 F5 3A 5C 37 FE 01 20 : 09
4478 05 CD 63 4B 01 C5 F1 6F : A6
---------------------------------
SUM: 63 CC 7D 76 E8 7C 8F 34 9E53

4480 26 00 3E 01 CD 99 4B CD : E3
4488 63 4B 02 FD 09 3A 5C 37 : 83
4490 FE 01 20 05 CD 63 4B 01 : A0
4498 C1 3E 00 32 5C 37 C9 B7 : 44
44A0 28 06 CD 63 4B 02 FD E1 : 89
44A8 C9 3A 15 44 F5 2A A3 5D : 7B
44B0 E5 CD CE 44 E1 22 A3 5D : C7
44B8 3A 15 44 47 F1 32 15 44 : 56
44C0 4F 78 91 47 20 02 0E 00 : CF
44C8 79 87 32 16 44 C9 3E 28 : BB
44D0 CD 83 56 CD AE 56 FE 29 : 9E
44D8 28 14 3A 15 44 F5 CD 9A : 2B
44E0 39 F1 3C 32 15 44 CD F3 : B1
44E8 44 CD 5C 34 38 EC 3E 29 : 2C
44F0 C3 83 56 CD 79 46 FE 01 : 27
44F8 20 1B CD C8 46 E5 21 FD : 19
---------------------------------
SUM: 75 9E 62 A1 73 5E 54 A0 0258

4500 36 CD 40 45 E1 E5 7D CD : 98
4508 A1 4B 21 FD 36 CD 59 45 : AB
4510 E1 7C C3 A1 4B FE 00 20 : 2A
4518 18 3A 91 4B FE EB 38 11 : 60
4520 CD C8 46 FD 2B 21 FD 73 : 94
4528 CD 40 45 21 FD 72 C3 59 : FE
4530 45 CD B5 45 21 FD 75 CD : 6C
4538 40 45 21 FD 74 C3 59 45 : 78
4540 CD 9C 4B 3A 15 44 3D 87 : 0B
4548 C6 70 FE 80 38 05 CD 60 : 1E
4550 61 3E 7E 32 63 45 C3 A1 : 5B
4558 4B CD 9C 4B 3A 63 45 3C : 1D
4560 C3 A1 4B 00 78 32 A2 5D : 58
4568 3E 28 CD 83 56 3A A2 5D : 45
4570 F5 2A A3 5D E5 FE 01 20 : 23
4578 05 CD AF 45 18 15 FE 02 : F3
---------------------------------
SUM: 29 BF E3 EA D2 5E F1 C1 E6B0

4580 20 05 CD BB 45 18 0C FE : 14
4588 03 20 05 CD CA 45 18 03 : 1F
4590 CD 1B 46 E1 22 A3 5D CD : FE
4598 17 44 F1 FE 04 38 05 CD : 58
45A0 57 46 18 06 B7 20 03 CD : 62
45A8 57 46 3E 29 C3 83 56 CD : 6D
45B0 E9 38 CD 9D 39 CD 7F 49 : 59
45B8 C3 C8 46 CD 9A 39 CD 16 : 54
45C0 46 CD 9D 39 CD E4 49 C3 : A6
45C8 C8 46 CD 9A 39 CD 16 46 : D7
45D0 CD 79 46 FE 00 28 2A CD : A9
45D8 9D 39 CD 16 46 CD 79 46 : 8B
45E0 FE 00 28 0F CD AF 45 CD : C3
45E8 63 4B 02 44 4D CD E9 38 : 2F
45F0 C3 C4 45 CD C4 45 CD 63 : D2
45F8 4B 02 E5 D5 CD AF 45 18 : E0
---------------------------------
SUM: 48 E6 43 DC 79 F7 6D 30 BFBE

4600 0C CD C8 46 CD B2 45 CD : 78
4608 16 46 CD B2 45 CD 63 4B : 9B
4610 04 44 4D D1 E1 C9 3E 2C : 7A
4618 C3 83 56 E5 F5 AF 32 66 : BD
4620 46 CD AE 56 FE 29 28 14 : 7A
4628 3A 66 46 F5 CD AF 45 CD : 69
4630 42 4B F1 3C 32 66 46 CD : 65
4638 5C 34 38 EC F1 B7 20 0D : 89
4640 26 00 3A 66 46 6F 3E 21 : DA
4648 CD 99 4B 18 08 47 3A 66 : B8
4650 46 B8 C4 C4 61 E1 C9 3A : CB
4658 66 46 B7 28 08 47 3E D1 : E9
4660 CD A1 4B 10 F9 C9 00 2A : B5
4668 EE 46 77 23 22 EE 46 C9 : ED
4670 2A EE 46 2B 7E 22 EE 46 : 5D
4678 C9 2A EE 46 2B 7E C9 2A : C3
---------------------------------
SUM: 54 22 4B 2F 51 21 67 5A B89E

4680 EE 46 2B 2B 7E C9 2A EE : E9
4688 46 2B 7E 2B BE C0 2A F0 : B2
4690 46 2B 5E 2B 56 2B 4E 2B : F4
4698 46 78 BA 20 02 79 BB C9 : 97
46A0 3E 01 E5 CD 67 46 E1 EB : 6A
46A8 2A F0 46 72 23 73 23 22 : AD
46B0 F0 46 C9 CD BC 46 E5 CD : 80
46B8 BC 46 D1 C9 2A F0 46 2B : 27
46C0 5E 2B 56 22 F0 46 EB C9 : EB
46C8 CD BC 46 E5 CD 70 46 E1 : 18
46D0 C9 2A F0 46 2B 5E 2B 56 : 33
46D8 2B 4E 2B 46 72 23 73 23 : 15
46E0 70 23 71 2A EE 46 2B 46 : D3
```

▶思わず"C"の袋とじと聞いて（見て）某エッチな本の仲間入りをしたのかと……。すいません，ごめんなさい。　棟方　正治(22)青森県

```
46E8 2B 4E 70 23 71 C9 00 CF : 15
46F0 20 CF E5 CD 79 46 FE 01 : 5F
46F8 20 12 CD BC 46 C1 79 C6 : 01
---------------------------------
SUM: CE 42 D0 DF 7C 69 FD D6 3078

4700 8E 4F 78 CE 69 47 CD 3A : DA
4708 47 C3 A7 46 CD 7F 49 E1 : 6D
4710 ED 5B 5B 63 19 C3 97 4B : C4
4718 01 AE 49 C3 21 47 01 E4 : 08
4720 49 E5 CD 5B 49 30 04 C1 : 94
4728 C3 3C 47 CD 3A 47 E1 C3 : 38
4730 10 47 CD 5B 49 DA 3C 47 : 25
4738 42 4B C5 C9 C5 CD 70 46 : 63
4740 CD B3 46 C1 79 C6 8E 4F : A3
4748 78 CE 69 47 CD 3A 47 C3 : 07
4750 A7 46 01 E8 DD 11 AF 47 : BA
4758 C3 32 47 01 EA DD 11 7C : 91
4760 47 C3 32 47 01 4F 00 11 : E4
4768 3F 48 C3 32 47 01 79 00 : 3D
4770 11 0C 49 C3 32 47 19 C9 : 84
4778 B7 ED 52 C9 CD 79 46 FE : 49
---------------------------------
SUM: 1E CB F0 7C 55 EC AC 08 4A2E

4780 01 28 23 CD E4 49 CD 63 : 76
4788 4B 03 B7 ED 52 3A 49 55 : 1C
4790 FE C2 20 04 3E 03 18 0A : 47
4798 FE CA 20 04 3E 02 18 02 : 46
47A0 3E 01 32 49 55 C9 CD BC : 61
47A8 46 CD 7B 6A CD A7 46 CD : 7F
47B0 CB 49 CD 79 46 FE 00 20 : BE
47B8 05 C3 14 48 18 10 FE 02 : 4C
47C0 20 05 C3 1D 48 18 07 FE : 6A
47C8 03 20 03 C3 1D 48 CD C8 : E3
47D0 46 7C B5 C8 7D D6 04 7C : 12
47D8 DE 00 30 06 45 0E 23 C3 : 4D
47E0 33 48 7D D6 FD 7C DE FF : 24
47E8 38 0A 7D ED 44 47 0E 2B : 70
47F0 C3 33 48 C9 2C 2D 20 19 : 99
47F8 7C FE 05 30 06 44 0E 24 : 2B
---------------------------------
SUM: 8D B5 9A A0 CC 7E 6C DB 42E3

4800 C3 33 48 7C FE FC 38 09 : F5
4808 7C ED 44 47 0E 25 C3 33 : 1D
4810 48 CD A0 46 CD AE 49 CD : 8C
4818 63 4B 01 19 C9 CD 86 46 : 2A
4820 20 0D CD C8 46 CD 7F 49 : 9D
4828 CD 63 4B 01 29 18 03 CD : 8D
4830 14 48 C9 C5 CD 7F 49 C1 : 40
4838 79 CD 30 3F 10 FA C9 CD : 55
4840 CB 49 CD 79 46 FE 01 20 : BF
4848 45 CD C8 46 7C B5 20 18 : 89
4850 CD 79 46 FE 00 20 08 CD : 7F
4858 63 4B 03 21 00 00 C9 CD : 68
4860 C8 46 21 00 00 C3 A0 46 : D8
4868 7C FE 00 20 03 7D FE 01 : 19
4870 C8 7D D6 11 7C DE 00 30 : B6
4878 0C E5 CD 7F 49 E1 AF 32 : 48
---------------------------------
SUM: BC 3D E0 7D 78 CC 9D 6E 455D

4880 C7 48 C3 97 48 CD EE 48 : B4
4888 DA C8 48 CD A0 46 CD AE : 18
4890 49 21 4F 00 C3 10 47 CB : 9E
4898 45 28 18 E5 21 C7 48 34 : CE
48A0 35 20 07 35 CD 63 4B 02 : 0E
48A8 54 5D E1 2B CD 97 48 3E : A7
48B0 19 18 11 CB 3C CB 1D 7C : AD
48B8 FE 00 20 03 7D FE 01 C4 : 61
48C0 97 48 3E 29 C3 A1 4B 00 : F5
48C8 C5 CD 7F 49 C1 78 FE 08 : 99
48D0 38 0D C5 CD 63 4B 03 65 : ED
48D8 2E 00 C1 78 D6 08 47 04 : 90
48E0 05 28 0A C5 CD 63 4B 01 : 78
48E8 29 C1 05 18 F2 C9 54 5D : 73
48F0 0E 00 06 10 7A FE 00 20 : BC
48F8 03 7B FE 01 20 03 41 37 : 18
---------------------------------
SUM: D0 74 E1 1C 35 46 6E 9B 0322

4900 C9 CB 3A CB 1B 38 03 0C : FB
4908 10 EA B7 C9 CD 79 46 FE : 04
4910 01 20 1A CD C8 46 7C B5 : 47
4918 CA DE 61 7C FE 00 20 03 : A6
```

```
4920 7D FE 01 C8 CD EE 48 DA : 21
4928 36 49 CD A0 46 CD E4 49 : 2C
4930 21 79 00 C3 10 47 C5 CD : 46
4938 7F 49 C1 78 FE 08 38 09 : 48
4940 C5 CD 83 3C C1 78 D6 08 : 68
4948 47 04 05 28 0D C5 CD 63 : 7A
4950 4B 04 CB 3C CB 1D C1 05 : 04
4958 18 EF C9 CD 79 46 FE 01 : 5B
4960 20 09 CD 7F 46 FE 01 20 : DA
4968 02 37 C9 B7 C9 CD 79 46 : 0E
4970 FE 00 20 09 CD 7F 46 FE : B7
4978 00 20 02 37 C9 B7 C9 CD : 6F
---------------------------------
SUM: 86 E0 CF 63 86 A2 F9 5D 2BA0

4980 70 46 FE 01 20 08 CD 42 : EC
4988 4B CD 6A 4A 18 1B FE 02 : FF
4990 20 08 CD 42 4B CD A4 4A : 3D
4998 18 0F FE 03 20 08 CD 42 : 5F
49A0 4B CD DE 4A 18 03 CD BC : E4
49A8 46 3E 00 C3 A2 46 CD CB : C7
49B0 49 CD 7F 46 FE 00 20 0D : 06
49B8 3A 91 4B FE EB 38 06 CD : 0A
49C0 D1 46 C3 E4 49 21 0F 4A : 81
49C8 C3 E7 49 CD 79 46 FE 02 : 7F
49D0 CC D1 46 CD 79 46 FE 00 : 6D
49D8 CC D1 46 CD 7F 46 FE 01 : 74
49E0 CC D1 46 C9 21 07 4A E5 : 03
49E8 CD 70 46 87 5F CD 70 46 : EC
49F0 87 4F E1 16 00 19 5E 23 : 67
49F8 56 EB 06 00 09 5E 23 56 : 27
---------------------------------
SUM: A9 DD E6 92 89 B7 40 22 E4A8

4A00 EB CD 81 1F C3 A9 49 1F : 2C
4A08 4A 27 4A 2F 4A 37 4A 17 : CC
4A10 4A 27 4A 2F 4A 37 4A 3F : F4
4A18 4A 64 4A 9E 4A D8 4A 47 : 49
4A20 4A 64 4A 9E 4A D8 4A 52 : 54
4A28 4A 72 4A AC 4A E4 4A 58 : 82
4A30 4A 8C 4A B5 4A ED 4A 5E : B4
4A38 4A 95 4A CF 4A F6 4A CD : 4F
4A40 63 4B 01 D1 C3 B3 46 CD : 09
4A48 70 4B CD 63 4B 01 E1 C3 : DB
4A50 B3 46 CD 10 4B C3 BC 46 : E6
4A58 CD 18 4B C3 BC 46 CD 24 : E6
4A60 4B C3 BC 46 CD BC 46 CD : AC
4A68 70 4B CD BC 46 3E 21 C3 : AC
4A70 99 4B CD 42 4B CD 8E 46 : DF
4A78 28 06 CD 10 4B C3 6A 4A : CD
---------------------------------
SUM: C0 C9 90 44 87 D5 5E AB D689

4A80 CD 6A 4A CD 63 4B 02 54 : 52
4A88 5D C3 BC 46 CD 42 4B CD : 49
4A90 18 4B C3 6A 4A CD 42 4B : 34
4A98 CD 24 4B C3 6A 4A CD BC : 3C
4AA0 46 CD 70 4B CD BC 46 3E : DB
4AA8 2A C3 99 4B CD 42 4B CD : F8
4AB0 10 4B C3 A4 4A CD 42 4B : 66
4AB8 CD 8E 46 28 06 CD 18 4B : FF
4AC0 C3 A4 4A CD A4 4A CD 63 : 9C
4AC8 4B 02 54 5D C3 BC 46 CD : 90
4AD0 42 4B CD 24 4B C3 A4 4A : 7A
4AD8 CD BC 46 CD 70 4B 21 FD : 75
4AE0 6E C3 27 4B CD 42 4B CD : CA
4AE8 10 4B C3 DE 4A CD 42 4B : A0
4AF0 CD 18 4B C3 DE 4A CD 42 : 2A
4AF8 4B CD 8E 46 28 06 CD 24 : 0B
---------------------------------
SUM: 0F A5 9A EF 0D AF 46 BE DD3A

4B00 4B C3 DE 4A CD DE 4A CD : F8
4B08 63 4B 02 54 5D C3 BC 46 : 26
4B10 CD BC 46 3E 11 C3 99 4B : C5
4B18 21 ED 5B CD 9C 4B CD BC : A6
4B20 46 C3 9C 4B 21 FD 5E E5 : 51
4B28 CD 9C 4B CD BC 46 CD E2 : 32
4B30 4B E1 F5 CD A1 4B 7C D6 : 2C
4B38 08 67 CD 9C 4B F1 3C C3 : 13
4B40 A1 4B 3A 67 63 FE 01 20 : 0F
4B48 07 3E 00 32 67 63 18 12 : 6B
4B50 3A 91 4B FE EB 38 06 FD : 3A
4B58 2B 3E D5 18 02 3E E5 CD : 48
4B60 A1 4B C9 D1 1A 47 13 1A : 14
4B68 13 CD A1 4B 10 F9 D5 C9 : 73
```

```
4B70 3A 91 4B FE EB 20 05 FD : 21
4B78 2B AF 18 11 FE EC 20 06 : 13
---------------------------------
SUM: 28 0E 51 04 6A 51 60 5C 09FB

4B80 FD 2B 3E 01 18 07 3E EB : AF
4B88 CD A1 4B 3E EB 32 91 4B : F0
4B90 C9 00 3E C3 C3 99 4B 3E : AF
4B98 CD CD A1 4B 7D CD A1 4B : BC
4BA0 7C D9 ED 5B 6A 1F B7 ED : CA
4BA8 52 38 1C CD F1 62 4D 65 : 78
4BB0 6D 6F 72 79 20 6F 76 65 : 31
4BB8 72 20 20 20 20 20 20 20 : 52
4BC0 20 20 0D 00 C3 77 30 FD : B4
4BC8 E5 D1 2A 5F 63 19 77 D9 : 0B
4BD0 FD 23 AF 32 91 4B C9 ED : 93
4BD8 5B 5F 63 19 5E 71 23 56 : 7E
4BE0 70 C9 3A 50 63 67 7D 94 : 9E
4BE8 C6 70 C9 CD E2 4B 6F FE : 66
4BF0 80 38 04 26 FF 18 02 26 : 21
4BF8 00 C9 CD D5 38 CD 8A 5F : 59
---------------------------------
SUM: 20 E6 20 D0 6F 92 60 C6 9F10

4C00 79 FE 3A 20 07 CD 68 52 : 5F
4C08 CD 76 4C D8 CD 8D 4C DA : E7
4C10 5C 4C CD 11 5C 3B 0D D8 : 02
4C18 CD 6F 5C 49 C6 03 4D 46 : 3D
4C20 4F D2 70 4E 57 48 49 4C : 13
4C28 C5 7D 50 52 45 50 45 41 : FF
4C30 D4 E3 50 4C 4F 4F D0 D2 : 93
4C38 50 45 58 49 D4 67 51 52 : 14
4C40 45 54 55 52 CE D9 4C 47 : 7A
4C48 4F 54 CF 22 52 43 41 53 : BD
4C50 C5 70 4D 00 38 03 21 22 : 00
4C58 39 C3 81 1F F5 CD AE 4C : 58
4C60 38 05 CD FA 4B 18 F6 F1 : 4E
4C68 B7 28 04 BD C4 00 61 C9 : 8E
4C70 21 8D 4C C3 79 4C 21 AE : 51
4C78 4C E5 CD AE 56 E1 DD E5 : A5
---------------------------------
SUM: 95 20 F3 42 E0 17 6E 50 064E

4C80 CD 81 1F DD E1 C9 CD 8D : 4E
4C88 4C D4 00 61 C9 CD 6F 5C : E2
4C90 7B 0D 01 00 42 45 47 49 : A0
4C98 CE 02 00 5B 0D 03 00 28 : 63
4CA0 0D 04 00 A2 0D 05 00 00 : C5
4CA8 38 02 2E 00 7D C9 CD BA : 35
4CB0 4C F5 FE 02 CC E4 4C F1 : 2E
4CB8 6F C9 CD 6F 5C 7D 0D 01 : 5B
4CC0 00 45 4E C4 02 00 5D 0D : C3
4CC8 03 00 29 0D 04 00 A3 0D : ED
4CD0 05 00 00 38 02 2E 00 7D : EA
4CD8 C9 CD AE 56 FE 28 CC 74 : 00
4CE0 39 CD F8 4C 3A 07 30 FE : B9
4CE8 00 20 07 CD 11 5C 3B 0D : A9
4CF0 18 05 3E 3B CD 83 56 C9 : 05
4CF8 3A 50 63 CD 9F 44 3E C9 : A4
---------------------------------
SUM: BE 7C DE 2C 68 8D 74 AE 2842

4D00 C3 A1 4B CD FA 54 CD 4B : E2
4D08 55 FD E5 CD 11 5C 54 48 : 0D
4D10 45 CE CD FA 4B CD 6F 5C : BD
4D18 45 4C 53 C5 00 00 45 4C : 3A
4D20 53 45 49 C6 01 00 45 C6 : B3
4D28 01 00 00 30 1B 7D 32 5F : 5A
4D30 4D CD 92 4B D1 CD 60 4D : 42
4D38 FD E5 3A 5F 4D B7 20 05 : A4
4D40 CD FA 4B 18 03 CD 03 4D : 4A
4D48 D1 CD 60 4D CD 6F 5C 45 : 28
4D50 4E 44 49 C6 00 00 46 C9 : B0
4D58 00 00 00 DC E4 4C C9 00 : D5
4D60 FD E5 E1 FD E5 1B 1B D5 : B0
4D68 FD E1 CD 9C 4B FD E1 C9 : 39
4D70 21 00 00 22 6E 4E CD D2 : 9E
4D78 52 CD 11 5C 4F C6 CD 86 : F4
---------------------------------
SUM: 99 4D 18 17 31 32 D0 03 8534

4D80 4C F5 CD E7 4D FD E5 CD : F1
4D88 5D 4E CD 76 4C 30 06 D1 : 41
4D90 CD 60 4D 18 24 2A 6E 4E : 9C
4D98 CD 92 4B FD E5 E1 2B 2B : C3
4DA0 22 6E 4E D1 CD 60 4D CD : F6
```

THE SENTINEL

▶Oh！Xはパソコンマガジンの域を脱して思想の世界へ入り込んでいる。あるいは，パソコン誌の名を借りて現代社会を見つめているのだろうか。前から感じていたけれど，最近特にそう思う。はっきりいってこの路線好きです。　石川　一彦(30)石川県

```
4DA8 11 5C 4F 54 48 45 52 D3 : C2
4DB0 30 05 CD 5D 4E 18 02 18 : DF
4DB8 C9 CD AE 4C F1 CD 68 4C : 02
4DC0 CD 6F 5C 45 4E 44 43 41 : F3
4DC8 53 C5 00 00 43 45 4E C4 : B2
4DD0 00 00 00 DC E4 4C 2A 6E : A4
4DD8 4E FD E5 C1 7C B5 28 06 : 50
4DE0 CD D7 4B EB 18 F6 C9 CD : 7E
4DE8 EF 38 E5 CD 11 5C 54 CF : 69
4DF0 30 19 E1 CD 33 4E CD 63 : A8
4DF8 4B 02 38 06 CD EF 38 CD : 4C
----------------------------------
SUM: 14 2C D4 AD 10 DB 92 60 97E7

4E00 4B 4E CD 63 4B 03 DA 00 : F1
4E08 00 18 21 E1 CD A0 46 CD : 9A
4E10 7F 49 CD E4 55 CD 5C 34 : 2B
4E18 30 0B CD 63 4B 02 28 06 : E6
4E20 CD EF 38 18 E7 CD 63 4B : 6E
4E28 03 C2 00 00 C9 01 7D 7C : 88
4E30 C3 36 4E 01 7B 7A EB 79 : A1
4E38 2E D6 63 CD 99 4B 78 2E : BE
4E40 DE 62 C3 99 4B 01 95 9C : 19
4E48 C3 4E 4E 01 93 9A EB 3E : B6
4E50 3E 6B 61 CD 99 4B 3E 3E : 37
4E58 6A 60 C3 99 4B CD 11 5C : AB
4E60 3A 0D 2A 6E 4E E5 CD FA : D9
4E68 4B E1 22 6E 4E C9 00 00 : D3
4E70 3E 02 CD 11 51 CD 11 5C : A9
4E78 28 0D 30 05 CD 85 4F 18 : 23
----------------------------------
SUM: EF EF EF 63 F8 B8 E3 57 C84D

4E80 03 CD 87 4E C3 02 51 CD : 88
4E88 80 5F CD DC 40 32 74 4F : BD
4E90 22 75 4F 3E 3D CD 83 56 : 07
4E98 CD 74 39 CD 6F 5C 54 CF : 35
4EA0 23 00 44 4F 57 4E 54 CF : 7E
4EA8 2B 00 00 D4 0A 62 7D 32 : 1A
4EB0 73 4F 3A 74 4F 2A 75 4F : AD
4EB8 CD A3 40 CD 92 4B FD E5 : 3C
4EC0 CD 0A 4F CD 7D 39 CD 79 : EF
4EC8 46 FE 00 20 1C CD 1C 4F : B8
4ED0 3E D2 CD 99 4B D1 FD E5 : 74
4ED8 D5 CD 60 4D CD 50 4F E1 : 9C
4EE0 CD FA 4F D1 CD 60 4D 18 : 79
4EE8 20 FD E1 CD DE 55 FD E5 : E0
4EF0 CD C8 46 E5 F5 CD 50 4F : 21
4EF8 CD 0A 4F F1 E1 CD A2 46 : AD
----------------------------------
SUM: AD 77 DB E0 23 F8 50 96 2FCA

4F00 CD 1C 4F E1 3E DA CD FC : FA
4F08 4F C9 CD D5 38 3A 74 4F : EF
4F10 2A 75 4F CD A2 46 3A 73 : 50
4F18 4F C3 8B 3E 3A 73 4F FE : D5
4F20 23 20 17 CD 79 46 FE 01 : E5
4F28 20 08 CD C8 46 CD 2D 4E : 4B
4F30 18 06 CD E4 49 CD E4 55 : 1E
4F38 18 15 CD 79 46 FE 01 20 : D8
4F40 08 CD C8 46 CD 45 4E 18 : 5B
4F48 06 CD E4 49 CD EC 55 C9 : D7
4F50 2A 75 4F E5 3A 74 4F 57 : 27
4F58 3A 73 4F 5F D5 CD 11 5C : 6A
4F60 44 CF CD 77 4F D1 7B 32 : 24
4F68 73 4F 7A 32 74 4F E1 22 : 34
4F70 75 4F C9 00 00 00 00 CD : 5A
4F78 FA 4B CD 11 5C 4E 45 58 : 6A
----------------------------------
SUM: A0 9A 9B 40 68 8B 7E 8D 5D80

4F80 D4 DC E4 4C C9 CD 32 50 : F8
4F88 FD E5 CD 3F 50 D1 FE 01 : 0E
4F90 20 05 CD 9B 4F 18 03 CD : C4
4F98 DE 4F C9 D5 CD 4B 55 FD : 35
4FA0 E5 CD 92 4B FD E5 CD 65 : A3
4FA8 50 C1 D1 E1 FE 01 20 05 : E7
4FB0 CD B9 4F 18 03 CD CE 4F : DA
4FB8 C9 D5 C5 C5 CD FA 4F D1 : 0F
4FC0 CD 60 4D CD 77 4F E1 CD : BB
4FC8 FA 4F D1 C3 60 4D D5 E5 : 44
4FD0 D5 FD E1 CD 77 4F E1 CD : F4
4FD8 FA 4F D1 C3 60 4D CD 92 : E9
4FE0 4B FD E5 CD 65 50 D1 FE : 7E
4FE8 01 20 06 D5 CD 60 4D 18 : 8E
4FF0 05 CD DE 55 FD E5 CD 77 : 2B
4FF8 4F E1 3E C3 47 EB 3A 18 : B5
----------------------------------
SUM: D0 F7 95 DE 24 66 1B 5B 75D6

5000 30 FE 01 20 28 FD E5 E1 : 3A
5008 23 23 B7 ED 52 7D D6 81 : 10
5010 7C DE 00 30 18 78 FE E0 : F8
5018 30 13 FE C3 20 04 3E 18 : 7E
5020 18 02 D6 A2 CD A1 4B 7D : C8
5028 ED 44 C3 A1 4B EB 78 C3 : 06
5030 99 4B CD AE 56 FE 3B C4 : B2
5038 28 39 3E 3B C3 83 56 CD : 43
5040 AE 56 FE 3B 28 15 CD FA : 41
5048 54 3A 4A 55 FE 01 20 07 : 53
5050 CD DE 55 3E 00 18 02 3E : 96
5058 01 18 02 3E 00 F5 3E 3B : C7
5060 CD 83 56 F1 C9 CD AE 56 : 31
5068 FE 29 28 07 CD 28 39 3E : C2
5070 01 18 02 3E 00 F5 3E 29 : B5
5078 CD 83 56 F1 C9 3E 03 CD : 6E
----------------------------------
SUM: 2E A9 CF 5F 68 4E A0 2F 4594

5080 11 51 FD E5 CD FA 54 3A : 99
5088 4A 55 FE 01 20 0F D1 CD : 6B
5090 DE 55 FD E5 CD B2 50 E1 : C5
5098 CD FA 4F 18 12 CD 4B 55 : AD
50A0 FD E5 CD B2 50 D1 E1 D5 : 38
50A8 CD FA 4F D1 CD 60 4D C3 : 24
50B0 02 51 CD 11 5C 44 CF CD : 6D
50B8 FA 4B CD 6F 5C 45 4E 44 : B4
50C0 57 48 49 4C C5 00 00 57 : 50
50C8 45 4E C4 00 00 00 DC E4 : 17
50D0 4C C9 3E 05 CD 11 51 FD : 84
50D8 E5 CD FA 4B E1 CD FA 4F : EE
50E0 C3 02 51 3E 04 CD 11 51 : 87
50E8 FD E5 CD FA 4B CD 11 5C : 2E
50F0 55 4E 54 49 CC D4 F3 61 : 34
50F8 CD FA 54 CD E4 4C E1 CD : C6
----------------------------------
SUM: 7B CB 08 D0 13 DA 28 48 1CBB

5100 4E 55 CD 2A 51 5E 23 56 : C2
5108 EB CD D9 4D 21 21 52 35 : A7
5110 C9 08 3A 21 52 FE 10 D2 : 5E
5118 3C 62 3C 32 21 52 CD 2A : 76
5120 51 08 36 00 23 36 00 23 : 0B
5128 77 C9 3A 21 52 5F 16 00 : 62
5130 21 37 51 19 19 19 C9 00 : BD
5138 00 00 00 00 00 00 00 00 : 00
5140 00 00 00 00 00 00 00 00 : 00
5148 00 00 00 00 00 00 00 00 : 00
5150 00 00 00 00 00 00 00 00 : 00
5158 00 00 00 00 00 00 00 00 : 00
5160 00 00 00 00 00 00 00 3E : 3E
5168 C3 CD A1 4B CD 6F 5C 54 : 68
5170 CF 85 51 46 52 4F CD D1 : 2A
5178 51 00 38 03 21 93 51 CD : 5E
----------------------------------
SUM: 0A E6 07 98 B3 CE AB DA A94A

5180 81 1F C3 E4 4C CD AE 56 : 64
5188 CD AE 5D FE 42 CA 28 62 : 6C
5190 C3 33 52 21 01 00 DD 7E : C5
5198 00 FE 28 CC EF 38 7C B5 : 4A
51A0 CA 28 62 7D D6 11 7C DE : 12
51A8 00 D2 28 62 2B 3A 21 52 : 34
51B0 F5 95 32 21 52 B7 20 05 : 0B
51B8 CD 28 62 18 0F FE 11 38 : C5
51C0 05 CD 28 62 18 06 CD 2A : 71
51C8 51 CD 11 52 F1 32 21 52 : 17
51D0 C9 CD 6F 5C 46 4F D2 02 : CA
51D8 00 57 48 49 4C C5 03 00 : FC
51E0 52 45 50 45 41 D4 04 00 : 45
51E8 4C 4F 4F D0 05 00 46 55 : 5A
51F0 4E C3 01 00 00 D2 D3 60 : 17
51F8 4D CD 2A 51 23 23 23 7C : 7A
----------------------------------
SUM: F5 97 72 A6 E4 E4 00 07 4C77

5200 FE 51 20 03 7D FE 37 CA : EE
5208 28 62 2B 7E 2B 2B B9 20 : 62
5210 EE FD E5 7E CD A1 4B 23 : 2A
5218 7E CD A1 4B D1 72 2B 73 : 18
5220 C9 00 3E C3 CD A1 4B CD : 50
5228 AE 56 CD AE 5D CD 33 52 : 2E
5230 C3 E4 4C B7 20 0D 3E 43 : 58
5238 CD D4 5C CD 58 52 21 00 : 95
5240 00 18 12 FE 43 20 07 E5 : 77
5248 CD 58 52 E1 18 07 FE 42 : B7
5250 28 03 CD 25 61 C3 9C 4B : 28
5258 FD E5 E1 3E 43 CD 26 36 : 6D
5260 2A 59 63 23 22 59 63 C9 : B0
5268 CD AE 5D B7 20 07 3E 42 : 36
5270 CD D4 5C 18 2E FE 43 20 : A4
5278 1E E5 FD E5 E1 3E 42 CD : 13
----------------------------------
SUM: 6D A3 AF 58 38 5C 30 82 A1BD

5280 26 36 E1 FD E5 C1 CD D7 : 84
5288 4B 2A 59 63 2B 22 59 63 : 3A
5290 EB 7C B5 20 F1 18 0C FE : 4F
5298 42 20 05 CD 37 61 18 03 : E7
52A0 CD 25 61 3E 3A C3 83 56 : 67
52A8 3E 28 CD 83 56 CD D2 52 : FD
52B0 3E 3A CD 83 56 CD 63 4B : 99
52B8 03 01 00 00 FD E5 21 4B : 52
52C0 00 CD 10 47 CD F5 52 60 : 98
52C8 69 D1 CD 63 4D 3E 29 C3 : E1
52D0 83 56 CD A9 49 CD 7A 39 : 18
52D8 CD 79 46 FE 00 20 05 CD : 7C
52E0 70 4B 18 03 CD E4 49 C9 : 99
52E8 3E 28 CD 83 56 CD F5 52 : 20
52F0 3E 29 C3 83 56 FD E5 CD : B2
52F8 6F 5C 22 0D 6E 53 5B 0D : 23
----------------------------------
SUM: FE E9 A9 F8 65 BF 9B 97 401F

5300 74 53 3C 0D 82 53 00 30 : 15
5308 05 CD 81 1F 18 20 CD 5C : D3
5310 36 FE 02 20 0D CD 3E 53 : C1
5318 CD 9C 4B 1B 7A B3 20 F8 : 14
5320 18 0C CD 3E 53 7D CD A1 : 6D
5328 4B 1B 7A B3 20 F7 CD 5C : D3
5330 34 38 C4 D1 FD E5 C1 79 : 1D
5338 93 4F 78 9A 47 C9 CD F2 : C3
5340 38 E5 CD 11 5C 3A 0D 30 : CE
5348 05 CD F2 38 18 03 21 01 : 39
5350 00 EB 7A B3 20 08 CD 8E : 9B
5358 61 11 01 00 18 0E 7B D6 : EA
5360 00 7A DE 80 38 06 CD 8E : 71
5368 61 11 01 00 E1 C9 CD 5E : 48
5370 54 FD 2B C9 CD E9 38 CD : 00
5378 74 39 CD C8 46 3E 5D C3 : E6
----------------------------------
SUM: 6D D7 9E D0 B0 5E F8 50 4322

5380 83 56 CD 80 5F CD AE 5D : 5D
5388 CD 33 52 3E 3E C3 83 56 : 6A
5390 3E 28 CD 83 56 CD A2 53 : CE
5398 CD 5C 34 38 F8 3E 29 C3 : B7
53A0 83 56 CD 6F 5C 53 54 52 : 6A
53A8 24 0D 83 01 46 4F 52 4D : E9
53B0 24 0D C9 01 00 3E 02 DA : 15
53B8 38 54 CD 6F 5C 50 4E 24 : E6
53C0 0D B5 01 44 45 43 49 24 : FC
53C8 0D C6 01 25 0D C6 01 48 : 15
53D0 45 58 32 24 0D 9C 01 48 : E5
53D8 45 58 34 24 0D 98 01 4D : E8
53E0 53 47 24 0D F8 01 4D 53 : 64
53E8 58 24 0D ED 01 21 0D ED : 92
53F0 01 43 48 52 24 0D 8D 01 : 9D
53F8 53 50 43 24 0D 7C 01 54 : E8
----------------------------------
SUM: 01 FA 2A 7A 7F B3 26 FC 86D7

5400 41 42 24 0D 78 01 43 52 : C2
5408 24 0D 80 01 00 3E 01 DA : CB
5410 38 54 DD 7E 00 FE 22 20 : 27
5418 0B DD 23 21 FC 01 CD 10 : 06
5420 47 C3 5E 54 FE 2F 20 07 : 10
5428 DD 23 21 94 01 18 06 CD : A1
5430 AF 45 21 C1 01 C3 10 47 : F1
5438 E5 F5 3E 28 CD 83 56 F1 : D7
5440 FE 01 20 05 CD AF 45 18 : FD
5448 0C FE 02 20 05 CD BB 45 : FE
5450 18 03 CD CA 45 3E 29 CD : 2B
5458 83 56 E1 C3 10 47 AF 32 : B5
5460 15 60 DD 7E 00 FE 20 30 : 1E
5468 05 CD 13 61 18 3C DD 23 : 9A
5470 FE 22 28 36 4F 3A 15 60 : 7C
5478 B7 20 25 79 FE 80 38 16 : 41
```

▶修学旅行が終わった。旅館に出現した体長ほんの1cmのゴキブリに群がる北海道人「おお，初めて見た。すげえ！」。帰りの飛行機内，離陸と同時にわき起こる歓声と拍手。苦笑いの一般客に「北海道へのフライトは初めてなんです」と意味不明なフォローをするスチュワーデス。これでも田舎者を自覚できた者は少ない。　渡辺　光(17)北海道

```
-----------------------------------
SUM: D4 67 8F BE CD C0 E1 8D 4350

5480 FE A0 30 07 3E 01 32 15 : 5B
5488 60 18 09 FE E0 38 05 3E : DA
5490 01 32 15 60 18 08 FE 5C : 22
5498 20 04 CD AE 54 4F 18 04 : 5E
54A0 AF 32 15 60 79 CD A1 4B : 88
54A8 18 B8 AF C3 A1 4B DD 7E : 89
54B0 00 FE 24 20 0F DD E5 D1 : E4
54B8 13 CD B5 1F D5 DD E1 DC : 23
54C0 13 61 18 35 CD 72 5C 5C : B8
54C8 0D 5C 00 22 0D 22 00 27 : E1
54D0 0D 27 00 4E 0D 0D 00 2F : CB
54D8 0D 0D 00 43 0D 0C 00 52 : C8
54E0 0D 1C 00 4C 0D 1D 00 55 : F4
54E8 0D 1E 00 44 0D 1F 00 30 : CB
54F0 0D 00 00 00 38 02 2E 5C : D1
54F8 7D C9 AF 32 49 55 CD D5 : 67
-----------------------------------
SUM: 37 97 7F 1F 17 A2 E8 E3 710B

5500 38 CD 7D 39 3E 00 32 4A : 75
5508 55 CD 79 46 FE 00 20 09 : 08
5510 3A 49 55 B7 C4 DE 55 18 : 9E
5518 2F 08 3E CA 32 49 55 08 : 17
5520 FE 01 20 18 CD BC 46 7C : 82
5528 B5 20 07 3E C3 32 49 55 : AD
5530 18 05 3E 01 32 4A 55 CD : FA
5538 A7 46 18 09 FE 03 20 05 : 34
5540 3E 04 32 49 55 CD 7F 49 : A7
5548 C9 00 00 21 FF FF E5 CD : 9A
5550 56 55 E1 C3 FC 4F 3A 49 : 1D
5558 55 B7 20 06 CD 1F 56 3E : B2
5560 CA C9 FE CA 20 06 CD 1F : 6D
5568 56 3E CA C9 FE C2 20 06 : 0D
5570 CD 1F 56 3E C2 C9 FE DA : E3
5578 20 06 CD E4 55 3E DA C9 : 0D
-----------------------------------
SUM: 27 93 24 48 44 6B B9 7B 94C5

5580 FE D2 20 06 CD E4 55 3E : 3A
5588 D2 C9 FE 01 20 06 CD E4 : 71
5590 55 3E CA C9 FE 02 20 03 : 49
5598 3E CA C9 FE 03 20 03 3E : 33
55A0 C2 C9 FE 06 20 06 CD EC : 6E
55A8 55 3E DA C9 FE 07 20 06 : 61
55B0 CD EC 55 3E D2 C9 FE 04 : E9
55B8 20 06 CD 09 56 3E CA C9 : 23
55C0 FE 05 20 06 CD 09 56 3E : 93
55C8 C2 C9 FE 08 20 06 CD F5 : 79
55D0 55 3E CA C9 FE 09 20 06 : 53
55D8 CD 00 56 3E CA C9 11 FD : 02
55E0 FF FD 19 C9 CD 63 4B 03 : 5C
55E8 B7 ED 52 C9 CD 63 4B 04 : 3E
55F0 7B 95 7A 9C C9 CD 63 4B : 6A
55F8 06 7C A2 67 7D A3 B4 C9 : 28
-----------------------------------
SUM: 80 A3 70 8E C9 37 FB 73 961B

5600 CD 63 4B 04 7C B5 B2 B3 : 15
5608 C9 11 FB FF FD 19 3E 7E : A6
5610 CD A1 4B FD 23 FD 23 3E : 37
5618 B6 CD A1 4B FD 23 C9 3A : 92
5620 91 4B FE 7C CA 77 56 FE : EB
5628 7A CA 77 56 FE 7B CA 77 : CB
5630 56 FE 7E CA 77 56 FE 7D : E4
5638 20 05 FD 23 C3 77 56 FE : D3
5640 03 20 0F 11 FC FF FD 19 : 54
5648 3E 7E CD A1 4B FD 23 C3 : 58
5650 7D 56 FE ED 20 0D 11 FC : F8
5658 FF FD 19 CD 63 4B 03 7E : 11
5660 23 B6 C9 FE EB 38 09 FD : C9
5668 2B CD 63 4B 02 7A B3 C9 : 9E
5670 CD 63 4B 02 7C B5 C9 CD : 44
5678 DE 55 CD A1 4B CD 63 4B : 67
-----------------------------------
SUM: 50 26 59 62 19 35 6C CD EB14

5680 01 B7 C9 F5 CD AE 56 F1 : 38
5688 DD BE 00 20 04 DD 23 18 : D7
5690 1C F5 CD F1 62 4D 69 73 : 5A
5698 73 69 6E 67 20 27 00 F1 : E9
56A0 CD F4 1F CD E2 1F 27 20 : F5
56A8 20 20 20 0D 00 C9 DD 7E : 91
56B0 00 FE 20 20 04 DD 23 18 : 5A
56B8 51 FE 0D 20 05 CD 82 5A : 2A
56C0 18 48 FE 2F 20 19 DD 7E : 21
56C8 01 FE 2F 20 05 CD 82 5A : FC
56D0 18 0B FE 2A 20 05 CD 24 : 61
56D8 57 18 02 18 2F 18 2B FE : F9
56E0 28 20 0C DD 7E 01 FE 2A : D8
56E8 20 22 CD 24 57 18 1B FE : BB
56F0 23 20 07 CD 68 57 30 14 : 1A
56F8 18 10 B7 20 0B 3A 67 59 : 04
-----------------------------------
SUM: B6 BE 34 06 FA 3E 92 0C BC6C

5700 B7 28 09 CD 03 59 18 02 : 2B
5708 18 02 18 A2 3A 53 58 FE : B7
5710 01 28 0D DD 34 00 DD 35 : 59
5718 00 28 05 CD 82 5A 18 8E : 7C
5720 DD 7E 00 C9 DD 7E 00 FE : 7D
5728 28 20 02 3E 29 32 67 57 : A1
5730 DD 23 DD 23 DD 7E 00 FE : 59
5738 0D 20 05 CD 82 5A 18 24 : 17
5740 B7 20 0B 3A 67 59 B7 28 : BB
5748 1D CD 03 59 18 16 FE 2A : 9C
5750 20 10 DD 23 3A 67 57 DD : 05
5758 BE 00 20 04 DD 23 18 06 : 00
5760 18 02 DD 23 18 CE C9 2F : F8
5768 CD 72 5C 23 49 C6 D9 57 : FD
5770 23 45 4C 53 C5 10 58 23 : 57
5778 45 4E 44 49 C6 25 58 00 : 63
-----------------------------------
SUM: BE 5F EB AC DA 50 5A 18 79CB

5780 38 2A 3A 53 58 FE 00 C8 : 0D
5788 CD 72 5C 23 43 48 41 49 : D3
5790 CE 55 58 23 49 4E 43 4C : C4
5798 55 44 C5 BE 58 23 45 4E : 2A
57A0 C4 B1 57 23 53 54 4F D0 : B5
57A8 BB 57 00 D0 CD 81 1F 37 : 86
57B0 C9 3E 00 32 54 58 DD 36 : F8
57B8 00 00 C9 CD F1 62 43 6F : 9B
57C0 6D 70 69 6C 65 20 73 74 : 1E
57C8 6F 70 20 21 21 20 20 20 : A1
57D0 20 20 20 20 0D 00 C3 77 : C7
57D8 30 3A 54 58 FE 01 20 1A : 4F
57E0 CD F1 62 23 49 46 20 6E : 60
57E8 65 73 74 69 6E 67 20 20 : CA
57F0 20 20 20 20 20 20 20 20 : 00
57F8 0D 00 CD F2 38 7C B5 20 : 55
-----------------------------------
SUM: FB 39 93 EC 41 D0 E2 4A F654

5800 04 3E 00 18 02 3E 01 32 : CD
5808 53 58 3E 01 32 54 58 C9 : 91
5810 CD 34 58 D0 3A 53 58 FE : 0C
5818 01 20 04 3E 00 18 02 3E : BB
5820 01 32 53 58 C9 CD 34 58 : 00
5828 D0 3E 00 32 54 58 3E 01 : 2B
5830 32 53 58 C9 3A 54 58 FE : 8A
5838 01 20 02 37 C9 CD F1 62 : 43
5840 4D 69 73 73 69 6E 67 20 : FA
5848 23 49 46 20 20 20 20 0D : 3F
5850 00 B7 C9 00 00 DD E5 D1 : 13
5858 3A 68 63 FE 01 20 33 CD : 24
5860 C4 1F CD E2 1F 41 72 65 : C9
5868 20 79 6F 75 20 72 65 61 : D5
5870 64 79 20 74 6F 20 43 48 : 8B
5878 41 49 4E 20 3F 20 00 CD : 24
-----------------------------------
SUM: 5C F8 D6 2D 05 C1 27 96 9350

5880 21 20 FE 1B CC 77 30 CD : 9A
5888 EB 1F DD E5 D1 CD EA 59 : AD
5890 18 06 DD E5 D1 CD A8 59 : 7F
5898 FD E5 E1 ED 4B 5F 63 09 : C6
58A0 ED 5B 70 1F 7D 93 7C 9A : FD
58A8 D8 2A 5B 63 09 EB 2A 70 : 4E
58B0 1F ED 4B 72 1F 09 7B 95 : 01
58B8 7A 9C D0 C3 49 61 3A 68 : F5
58C0 63 FE 01 CA D8 62 3A 67 : 07
58C8 59 FE 08 D2 D8 62 CD 57 : 8F
58D0 59 3A 5D 1F 67 3A 7F 5A : 89
58D8 6F CD FC 58 2A 80 5A CD : 61
58E0 FC 58 2A D7 5A CD FC 58 : D0
58E8 2A 65 63 CD FC 58 DD E5 : D5
58F0 D1 CD A8 59 3A 67 59 3C : D5
58F8 32 67 59 C9 7D 12 13 7C : D9
-----------------------------------
SUM: 2C 2C 6F 62 F5 74 A5 69 6A1B

5900 12 13 C9 CD E2 1F 49 6E : 73
5908 63 6C 75 64 65 20 65 6E : 00
5910 64 20 0D 00 3A 67 59 3D : C8
5918 32 67 59 CD 57 59 CD 50 : 8C
5920 59 7C 32 5D 1F 7D 32 7F : B1
5928 5A CD 50 59 22 80 5A CD : 99
5930 50 59 22 D7 5A CD 50 59 : 72
5938 22 65 63 CD DA 59 ED 5B : 32
5940 80 5A 2A 0C 30 3E 10 CD : 5B
5948 00 20 DC 96 30 C3 82 5A : 61
5950 1A 6F 13 1A 67 13 C9 3A : 33
5958 67 59 07 07 07 11 68 59 : A7
5960 83 5F 7A CE 00 57 C9 00 : 4A
5968 00 00 00 00 00 00 00 00 : 00
5970 00 00 00 00 00 00 00 00 : 00
5978 00 00 00 00 00 00 00 00 : 00
-----------------------------------
SUM: B4 AE 45 E9 1B 9E 29 23 8930

5980 00 00 00 00 00 00 00 00 : 00
5988 00 00 00 00 00 00 00 00 : 00
5990 00 00 00 00 00 00 00 00 : 00
5998 00 00 00 00 00 00 00 00 : 00
59A0 00 00 00 00 00 00 00 00 : 00
59A8 3E 04 CD A3 1F CD 09 20 : C7
59B0 DC 96 30 CD DA 59 CD 1E : 8D
59B8 5A 2A 74 1F 7D C6 1E 6F : E7
59C0 7C CE 00 67 7E 32 7F 5A : 3A
59C8 CD 4F 5A 2A 0C 30 22 70 : 6E
59D0 1F 21 00 10 22 72 1F C3 : C6
59D8 30 5A ED 5B 5E 1F 2A 62 : DB
59E0 1F 3E 01 CD 00 20 DC 96 : BD
59E8 30 C9 3E 04 CD A3 1F CD : 97
59F0 09 20 DC 96 30 28 14 CD : D4
59F8 E2 1F 46 6F 75 6E 64 20 : 1D
-----------------------------------
SUM: 46 A2 19 61 F2 38 51 EC E446

5A00 20 20 00 CD 9D 1F CD EB : 81
5A08 1F 18 E4 CD 1E 5A 2A 0E : 98
5A10 30 22 70 1F CD 30 5A CD : 05
5A18 A6 1F DC 96 30 C9 CD E2 : DF
5A20 1F 4C 6F 61 64 69 6E 67 : DD
5A28 20 00 CD 9D 1F C3 EB 1F : 76
5A30 2A 0C 30 3A 68 63 FE 01 : 6A
5A38 20 03 2A 0E 30 22 65 63 : 75
5A40 21 00 00 22 D7 5A DD 21 : 72
5A48 11 5B DD 36 00 0D C9 2A : 7F
5A50 62 1F 3A 7F 5A 5F 16 00 : 09
5A58 19 7E B7 20 05 3E 07 CD : 85
5A60 96 30 FE 80 38 01 AF 32 : 5E
5A68 7F 5A EB 29 29 29 29 22 : 8A
5A70 80 5A EB 2A 0C 30 3E 10 : 79
5A78 CD 00 20 DC 96 30 C9 00 : 58
-----------------------------------
SUM: AD B0 88 3B 0C B1 7C 0E D495

5A80 00 00 2A 65 63 11 11 5B : 6F
5A88 06 00 04 04 05 CA A8 62 : E7
5A90 C5 CD EB 5A C1 12 B7 28 : 89
5A98 05 13 FE 0D 20 EC 22 65 : B6
5AA0 63 2A D7 5A 23 22 D7 5A : 34
5AA8 3A D6 5A FE 00 CC B5 5A : 43
5AB0 DD 21 11 5B C9 3A 11 5B : D9
5AB8 B7 28 15 FD E5 E1 CD BE : 42
5AC0 1F CD D9 5A CD F1 1F 11 : 0D
5AC8 11 5B CD E8 1F CD EB 1F : 17
5AD0 CD C7 1F 77 30 C9 00 00 : 23
5AD8 00 2A D7 5A 11 E5 5A CD : 78
5AE0 97 6B C3 E5 1F 31 32 33 : 5F
5AE8 34 35 00 3A 68 63 FE 00 : 6C
5AF0 20 1C D5 ED 5B 0C 30 7B : 10
5AF8 C6 00 5F 7A CE 10 57 7C : 50
-----------------------------------
SUM: AF FE 01 19 F7 FE 17 3E 5992

5B00 BA 20 02 7D BB 20 06 CD : 07
5B08 4F 5A 2A 0C 30 D1 7E 23 : 81
5B10 C9 00 00 00 00 00 00 00 : C9
5B18 00 00 00 00 00 00 00 00 : 00
5B20 00 00 00 00 00 00 00 00 : 00
5B28 00 00 00 00 00 00 00 00 : 00
5B30 00 00 00 00 00 00 00 00 : 00
5B38 00 00 00 00 00 00 00 00 : 00
```

THE SENTINEL

▶富山では結婚式にはでっけーかまぼこをたくさんもらうのが常識ですが，私は24時間テレビを見るまでは，全国的に結婚式にはかまぼこがつきものだと思っていた。
杉田 正章(17)富山県

```
5B40 00 00 00 00 00 00 00 00 : 00
5B48 00 00 00 00 00 00 00 00 : 00
5B50 00 00 00 00 00 00 00 00 : 00
5B58 00 00 00 00 00 00 00 00 : 00
5B60 00 00 00 00 00 00 00 00 : 00
5B68 00 00 00 00 00 00 00 00 : 00
5B70 00 00 00 00 00 00 00 00 : 00
5B78 00 00 00 00 00 00 00 00 : 00
---------------------------------
SUM: D2 7A 2C 89 EB F1 84 F0 A2FD

5B80 00 00 00 00 00 00 00 00 : 00
5B88 00 00 00 00 00 00 00 00 : 00
5B90 00 00 00 00 00 00 00 00 : 00
5B98 00 00 00 00 00 00 00 00 : 00
5BA0 00 00 00 00 00 00 00 00 : 00
5BA8 00 00 00 00 00 00 00 00 : 00
5BB0 00 00 00 00 00 00 00 00 : 00
5BB8 00 00 00 00 00 00 00 00 : 00
5BC0 00 00 00 00 00 00 00 00 : 00
5BC8 00 00 00 00 00 00 00 00 : 00
5BD0 00 00 00 00 00 00 00 00 : 00
5BD8 00 00 00 00 00 00 00 00 : 00
5BE0 00 00 00 00 00 00 00 00 : 00
5BE8 00 00 00 00 00 00 00 00 : 00
5BF0 00 00 00 00 00 00 00 00 : 00
5BF8 00 00 00 00 00 00 00 00 : 00
---------------------------------
SUM: 00 00 00 00 00 00 00 00 0000

5C00 00 00 00 00 00 00 00 00 : 00
5C08 00 00 00 00 00 00 00 00 : 00
5C10 00 CD AE 56 DD E5 E1 D1 : 45
5C18 1A 13 FE 0D 28 2E FE 80 : 0C
5C20 38 14 D6 80 CD 55 5C 20 : 40
5C28 0B AF 32 15 60 23 7E CD : CF
5C30 B9 5F 30 18 18 1C CD 55 : B6
5C38 5C 28 0E 1A 13 FE 20 38 : 15
5C40 06 FE 80 30 02 18 F4 18 : DA
5C48 09 23 18 CC E5 DD E1 37 : EA
5C50 D5 C9 B7 D5 C9 47 7E CD : 85
5C58 66 5C B8 C9 08 3A 15 60 : FA
5C60 B7 28 02 08 C9 08 FE 61 : 19
5C68 D8 FE 7B D0 C6 E0 C9 CD : 5D
5C70 AE 56 D1 DD E5 E1 1A FE : 90
5C78 0D 28 37 FE 80 38 14 D6 : 0C
---------------------------------
SUM: 06 14 7E 77 09 1C 03 49 308F

5C80 80 CD 55 5C 20 0B AF 32 : 0A
5C88 15 60 23 7E CD B9 5F 30 : 2B
5C90 21 18 09 CD 55 5C 20 04 : E4
5C98 23 13 18 DA 1A 13 FE 0D : 60
5CA0 28 06 FE 80 30 02 18 F4 : EA
5CA8 13 13 1A B7 20 C5 13 B7 : A6
5CB0 D5 C9 E5 DD E1 13 1A 6F : DD
5CB8 13 1A 67 13 1A B7 28 10 : B0
5CC0 1A 13 FE 0D 28 06 FE 80 : E4
5CC8 30 02 18 F4 13 13 18 EC : 68
5CD0 13 37 D5 C9 F5 CD 80 5F : 89
5CD8 3E 01 32 A5 5D FD E5 E1 : 36
5CE0 F1 32 A1 5D 22 9F 5D 3A : 79
5CE8 A5 5D FE 00 20 16 CD 64 : 67
5CF0 5D 11 48 63 2A 48 63 ED : DB
5CF8 4B 46 63 7D 91 7C 98 D4 : EA
---------------------------------
SUM: D5 87 64 54 31 20 39 A8 3C58

5D00 6F 62 18 11 11 4C 63 2A : E4
5D08 4C 63 ED 4B 4A 63 7D 91 : A2
5D10 7C 98 D4 8C 62 AF 32 15 : CC
5D18 60 06 00 DD 7E 00 CD 5C : EA
5D20 5C CD B9 5F 30 0F CD 9A : E7
5D28 1F 04 78 FE 21 D2 C0 62 : AE
5D30 DD 23 23 18 E6 ED 4B 9F : F8
5D38 5D 3E 0D CD 9A 1F 23 79 : CA
5D40 CD 9A 1F 23 78 CD 9A 1F : A7
5D48 23 3A A1 5D CD 9A 1F 23 : 04
5D50 3A A2 5D CD 9A 1F 23 AF : 91
5D58 CD 9A 1F 7D 12 13 7C 12 : B6
5D60 22 A3 5D C9 CD 59 5E CD : 3C
5D68 94 1F 5F 23 CD 94 1F 57 : 0C
5D70 23 7A B3 28 1C ED 5B 51 : 2D
5D78 63 7D 93 7C 9A 38 03 2A : EE
---------------------------------
SUM: 7F 5E 78 61 4D F6 0D E2 4C67
```

```
5D80 12 30 ED 5B 8E 5E 7C BA : AC
5D88 20 02 7D BB CC 54 62 18 : F4
5D90 D6 ED 5B 48 63 2B 7A CD : 3B
5D98 9A 1F 2B 7B C3 9A 1F 00 : DB
5DA0 00 00 00 00 00 00 3E 00 : 3E
5DA8 32 A5 5D C3 B3 5D 3E 01 : 46
5DB0 32 A5 5D 3A A5 5D FE 00 : 6E
5DB8 CA D5 5D 2A 10 30 CD 94 : C7
5DC0 1F B7 C8 CD 0D 5E D8 CD : 7B
5DC8 94 1F 23 FE 0D 20 F8 23 : 1C
5DD0 23 23 23 18 E9 CD 59 5E : EE
5DD8 44 4D 60 69 CD 94 1F 5F : 39
5DE0 23 CD 94 1F 57 EB 7C B5 : 16
5DE8 28 20 CD 0D 5E D8 03 03 : 5E
5DF0 ED 5B 51 63 79 93 78 9A : 1A
5DF8 38 04 ED 4B 12 30 ED 5B : FE
---------------------------------
SUM: 5A EF 14 26 F8 C6 EA 8E 86F5

5E00 8E 5E 78 BA 20 02 79 BB : 74
5E08 20 D0 AF B7 C9 DD E5 D1 : B2
5E10 AF 32 15 60 CD 94 1F FE : D4
5E18 0D 20 08 1A CD B9 5F 30 : 64
5E20 10 B7 C9 CD 50 5E 28 02 : 35
5E28 B7 C9 CD B9 5F 23 13 18 : B3
5E30 E3 D5 DD E1 23 CD 94 1F : 19
5E38 5F 23 CD 94 1F 57 23 CD : 49
5E40 94 1F 4F 23 CD 94 1F 47 : EC
5E48 23 22 A3 5D EB 79 37 C9 : A9
5E50 C5 47 1A CD 5C 5C B8 C1 : 24
5E58 C9 DD E5 D1 01 00 00 AF : 0C
5E60 32 15 60 1A CD 5C 5C CD : 13
5E68 B9 5F 30 10 60 69 29 29 : 73
5E70 29 09 4F 06 00 09 29 44 : FD
5E78 4D 13 18 E7 60 69 ED 5B : 70
---------------------------------
SUM: 19 ED 6C 1B 16 71 77 D5 7581

5E80 53 63 CD 65 6A ED 5B 12 : AC
5E88 30 19 22 8E 5E C9 00 00 : 20
5E90 2A 12 30 ED 5B 51 63 7C : E4
5E98 BA 20 02 7D BB 28 07 AF : F2
5EA0 CD 9A 1F 23 18 F1 AF CD : 2E
5EA8 9A 1F 2A 14 30 22 48 63 : F4
5EB0 C9 2A 14 30 3A A5 5D FE : 71
5EB8 01 20 03 2A 10 30 CD 94 : EF
5EC0 1F B7 C8 E5 CD 94 1F 23 : 26
5EC8 FE 0D 20 F8 23 23 CD 94 : CA
5ED0 1F 23 23 D1 FE 42 20 05 : 9B
5ED8 CD E6 5E 18 07 FE C0 38 : 26
5EE0 03 CD E6 5E 18 D8 E5 EB : D4
5EE8 CD 94 1F FE 0D 28 06 CD : 86
5EF0 F4 1F 23 18 F3 CD EB 1F : 18
5EF8 E1 C9 2A 7E 5F CD 94 1F : 31
---------------------------------
SUM: 46 C7 3C A6 DC A8 1C E9 EEA4

5F00 B7 C8 E5 CD 94 1F 23 FE : 05
5F08 0D 20 F8 CD 94 1F 4F 23 : 17
5F10 CD 94 1F 47 23 CD 94 1F : 6A
5F18 23 23 FE 80 38 0B CD E2 : B6
5F20 1F 46 75 6E 63 20 00 18 : E3
5F28 27 FE 41 20 0B CD E2 1F : 5F
5F30 43 6F 6E 73 74 00 18 18 : 37
5F38 FE 21 38 0B CD E2 1F 41 : 71
5F40 72 72 61 79 00 18 09 CD : AC
5F48 E2 1F 56 61 72 20 20 00 : 6A
5F50 D1 CD 5B 5F CD C7 1F 77 : 82
5F58 30 18 A2 E5 3E 20 CD F4 : EE
5F60 1F 60 69 CD BE 1F 3E 3A : 0A
5F68 CD F4 1F EB CD 94 1F FE : 49
5F70 0D 28 06 CD F4 1F 23 18 : 56
5F78 F3 CD EB 1F E1 C9 00 00 : 74
---------------------------------
SUM: 7C 32 83 2F 0F 9F 81 3A CEFC

5F80 CD 8A 5F D8 CD D3 60 C3 : 51
5F88 77 30 AF 32 15 60 CD AE : 78
5F90 56 DD E5 D1 CD C4 5F 38 : 11
5F98 04 AF 0E 00 C9 13 1A CD : 84
5FA0 B9 5F 38 F9 4F FE 28 20 : DE
5FA8 04 3E 02 18 0A FE 5B 20 : DF
5FB0 04 3E 03 18 02 3E 01 37 : D5
5FB8 C9 CD C4 5F D8 FE 3A D0 : 99
5FC0 FE 30 3F C9 08 3A 15 60 : ED
5FC8 B7 28 0B AF 32 15 60 08 : 48
```

```
5FD0 FE 40 38 3F 18 3B 08 FE : 0E
5FD8 40 38 38 FE 5B 38 32 FE : 71
5FE0 5E 38 30 FE 7B 28 2C FE : 91
5FE8 7D 28 28 FE 80 38 22 FE : A3
5FF0 A0 30 09 08 3E 01 32 15 : 67
5FF8 60 08 18 15 FE E0 38 09 : B4
---------------------------------
SUM: F6 56 35 31 8F 45 CB 3B 0823

6000 08 3E 01 32 15 60 08 18 : 0E
6008 08 FE A2 28 06 FE A3 28 : 9F
6010 02 37 C9 B7 C9 00 21 00 : A3
6018 00 16 00 DD 7E 00 FE 5F : CE
6020 28 0C CD 66 5C CD B8 1F : 67
6028 30 04 FD E5 E1 C9 DD 7E : 1B
6030 00 FE 5F 20 04 DD 23 18 : 99
6038 F5 CD 66 5C CD B8 1F D8 : 00
6040 DD 23 29 29 29 29 5F 19 : 1C
6048 18 E4 DD E5 E1 23 7E CD : 0D
6050 BF 60 38 F9 7E CD 66 5C : 5D
6058 FE 48 20 06 CD 16 60 DD : 8C
6060 23 C9 2B 7E FE 5F 28 FA : 14
6068 CD 66 5C FE 42 20 2E 21 : 3E
6070 00 00 16 00 DD 7E 00 FE : 6F
6078 5F 20 04 DD 23 18 F5 FE : 8E
---------------------------------
SUM: 60 62 FA 1B 05 CD 8F 62 4952

6080 30 20 03 B7 18 09 FE 31 : 5A
6088 20 03 37 18 02 18 06 ED : 7F
6090 6A DD 23 18 DF E5 3E 42 : C6
6098 CD 83 56 E1 C9 21 00 00 : 71
60A0 16 00 DD 7E 00 FE 5F 20 : EE
60A8 04 DD 23 18 F5 D6 30 D8 : EF
60B0 FE 0A D0 29 44 4D 29 29 : E4
60B8 09 5F 19 DD 23 18 E3 FE : 7A
60C0 5F 20 02 37 C9 CD BD 5F : 6A
60C8 D8 CD 66 5C FE 41 3F D0 : B5
60D0 FE 47 C9 E5 CD F1 62 53 : 66
60D8 79 6E 74 61 78 20 65 72 : 2B
60E0 72 6F 72 0D 00 E1 C9 CD : D7
60E8 F1 62 49 6C 6C 65 67 61 : A1
60F0 6C 20 63 6F 6E 73 74 61 : 14
60F8 6E 74 0D 00 21 00 00 C9 : D9
---------------------------------
SUM: 93 D0 6C 25 25 38 44 CB 9216

6100 CD F1 62 49 6C 6C 65 67 : 0D
6108 61 6C 20 62 72 61 63 65 : EA
6110 0D 00 C9 CD F1 62 42 61 : 99
6118 64 20 73 74 72 69 6E 67 : 1B
6120 20 20 0D 00 C9 CD F1 62 : 36
6128 49 6C 6C 65 67 61 6C 20 : DA
6130 6E 61 6D 65 0D 00 C9 CD : 44
6138 F1 62 44 75 70 20 64 65 : 65
6140 66 20 6E 61 6D 65 0D 00 : 34
6148 C9 CD F1 62 49 6C 6C 65 : 6F
6150 67 61 6C 20 61 64 64 72 : EF
6158 65 73 73 0D 00 C3 77 30 : C2
6160 CD F1 62 54 6F 6F 20 6D : DF
6168 61 6E 79 20 61 72 67 75 : 17
6170 6D 65 6E 74 73 0D 00 C9 : FD
6178 CD F1 62 54 6F 6F 20 6D : DF
---------------------------------
SUM: CA 42 D1 57 B7 3B FD 67 0155

6180 61 6E 79 20 64 61 74 61 : 02
6188 20 20 20 0D 00 C9 CD F1 : F4
6190 62 4F 75 74 20 6F 66 20 : AF
6198 72 61 6E 67 65 20 20 20 : 6D
61A0 20 0D 00 21 00 00 C9 CD : E4
61A8 F1 62 4C 6F 63 61 6C 20 : 5E
61B0 61 72 69 61 20 6F 76 65 : 07
61B8 72 66 6C 6F 77 0D 00 21 : 58
61C0 00 00 AF C9 CD F1 62 55 : ED
61C8 6E 6D 61 74 63 68 65 64 : 44
61D0 20 61 72 67 75 6D 65 6E : 0F
61D8 74 73 20 0D 00 C9 CD F1 : 9B
61E0 62 44 65 76 20 62 79 20 : 9C
61E8 30 20 20 20 20 20 20 20 : 10
61F0 0D 00 C9 CD F1 62 4D 69 : AC
61F8 73 73 69 6E 67 20 55 4E : E7
---------------------------------
SUM: 4D 9D F6 EA 20 29 A6 14 20DC

6200 54 49 4C 20 20 20 20 0D : 76
```

▶初めて嫁さんネタ以外でハガキが載った。さて，68000円でまんまとだまされた嫁さんは(ばれたケド)，このあいだな～んと98を買ってくれた。しかし！　それは自分がX68000を独占するためだった。どうやらこの家の主人はこの女の人だと判断したX68000は嫁さん相手に絶好調である。

遠藤　勇(33)大阪府

```
6208 00 C9 CD F1 62 4D 69 73 : 12
6210 73 69 6E 67 20 54 4F 2F : A3
6218 44 4F 57 4E 54 4F 20 20 : 1B
6220 20 20 0D 00 21 23 00 C9 : 5A
6228 CD F1 62 43 61 6E 27 74 : CD
6230 20 6A 75 6D 70 20 20 20 : 3C
6238 20 0D 00 C9 CD F1 62 4E : 64
6240 65 73 74 69 6E 67 20 6F : 19
6248 76 65 72 66 6C 6F 77 0D : 12
6250 00 C3 77 30 CD F1 62 48 : D2
6258 61 73 68 20 74 61 62 6C : FF
6260 65 20 6F 76 65 72 66 6C : 13
6268 6F 77 0D 00 C3 77 30 CD : 2A
6270 F1 62 47 6C 6F 62 61 6C : A4
6278 20 74 61 62 6C 65 20 6F : B7
---------------------------------
SUM: 59 CD AB A2 D3 8A 13 BE CAF0

6280 76 65 72 66 6C 6F 77 0D : 12
6288 00 C3 77 30 CD F1 62 4C : D6
6290 6F 63 61 6C 20 74 61 62 : F6
6298 6C 65 20 6F 76 65 72 66 : 13
62A0 6C 6F 77 0D 00 C3 77 30 : C9
62A8 CD F1 62 54 6F 6F 20 6C : DE
62B0 6F 6E 67 20 6C 69 6E 65 : 0C
62B8 20 20 20 0D 00 C3 77 30 : D7
62C0 CD F1 62 54 6F 6F 20 6C : DE
62C8 6F 6E 67 20 6E 61 6D 65 : 05
62D0 20 20 20 0D 00 C3 77 30 : D7
62D8 CD F1 62 43 61 6E 27 74 : CD
62E0 20 49 4E 43 4C 55 44 45 : 24
62E8 20 20 20 20 0D 00 C3 77 : C7
62F0 30 CD C4 1F AF CD 33 20 : AF
62F8 2A 55 63 23 22 55 63 ED : CC
---------------------------------
SUM: DC D9 AA 68 12 0F F0 90 C0CA

6300 5B 16 30 7D 93 7C 9A 38 : FF
6308 19 CD E2 1F 54 6F 6F 20 : 39
6310 6D 61 6E 79 20 65 72 72 : 1E
6318 6F 72 73 21 21 0D 00 C3 : 66
6320 77 30 3A D6 5A FE 00 C4 : D3
6328 B5 5A DD E5 E1 11 F9 A4 : 60
6330 19 3E 20 CD F4 1F 2B 7C : FE
6338 B5 20 F6 3E 5E CD F4 1F : 47
6340 CD EB 1F C3 E2 1F 00 00 : 9B
6348 00 00 00 00 00 00 00 00 : 00
6350 00 00 00 00 00 00 00 00 : 00
6358 00 00 00 00 00 00 00 00 : 00
6360 00 00 00 00 00 00 00 00 : 00
6368 00 00 00 00 00 00 00 00 : 00
6370 00 00 00 00 00 00 00 00 : 00
6378 00 00 00 00 00 00 00 00 : 00
---------------------------------
SUM: 17 89 3F BF 97 77 93 90 E328

6380 00 00 00 00 00 00 00 00 : 00
6388 00 00 00 00 00 00 00 00 : 00
6390 00 00 00 00 00 00 00 00 : 00
6398 00 00 00 00 00 00 00 00 : 00
63A0 00 00 00 00 00 00 00 00 : 00
63A8 00 00 00 00 00 00 00 00 : 00
63B0 00 00 00 00 00 00 00 00 : 00
63B8 00 00 00 00 00 00 00 00 : 00
63C0 00 00 00 00 00 00 00 00 : 00
63C8 00 00 00 00 00 00 00 00 : 00
63D0 00 00 00 00 00 00 00 00 : 00
63D8 00 00 00 00 00 00 00 00 : 00
63E0 00 00 00 00 00 00 00 00 : 00
63E8 00 00 00 00 00 00 00 00 : 00
63F0 00 00 00 00 00 00 00 00 : 00
63F8 00 00 00 00 00 00 00 DD : DD
---------------------------------
SUM: 00 00 00 00 00 00 00 DD 00DD

6400 E5 AF 32 A2 5D DD 21 E0 : A3
6408 66 11 00 00 CD 2E 64 DD : B3
6410 21 AF 67 ED 5B 5B 63 CD : 0A
6418 2E 64 2A 48 63 22 7E 5F : 66
6420 DD 21 D7 66 ED 5B 5B 63 : 41
6428 CD 2E 64 DD E1 C9 3E 00 : 24
6430 32 A5 5D DD 46 00 DD 23 : 57
6438 78 B7 C8 FE 81 20 07 DD : 7A
6440 7E 00 DD 23 18 01 AF 32 : 78
6448 A2 5D DD 6E 00 DD 23 DD : 27
6450 66 00 DD 23 19 D5 78 CD : 99
6458 E1 5C D1 DD 23 18 D4 21 : 1B
6460 8E 69 CD 48 65 D4 73 64 : 1C
6468 ED 4B 0A 30 79 95 78 9C : 94
6470 30 F0 C9 7E FE 40 38 05 : E2
6478 FE C0 DA 0E 65 FE DD CA : B0
---------------------------------
SUM: FE 9B 05 8A 12 3E 01 18 CDE2

6480 B7 64 FE FD CA B7 64 FE : F9
6488 ED CA A8 64 E6 07 FE 06 : B4
6490 CA 0B 65 11 1C 65 CD 13 : AC
6498 65 DA D8 64 11 37 65 CD : F5
64A0 13 65 DA 0B 65 C3 0E 65 : F8
64A8 CD 0E 65 11 41 65 CD 13 : D7
64B0 65 DA D8 64 C3 0E 65 CD : 7E
64B8 0E 65 7E FE 30 DA 73 64 : D0
64C0 FE 38 DA D2 64 FE 40 DA : 5E
64C8 73 64 FE E0 DA D2 64 C3 : 88
64D0 73 64 CD 73 64 C3 0E 65 : B1
64D8 CD 0E 65 5E 23 56 23 7B : B5
64E0 D6 8E 7A DE 69 38 1C ED : 66
64E8 4B 0A 30 79 93 78 9A 38 : DB
64F0 12 ED 4B 5B 63 79 D6 8E : E5
64F8 4F 78 DE 69 47 7B 81 5F : B0
---------------------------------
SUM: 59 D0 55 F2 E1 F7 29 1C 6722

6500 7A 88 57 7B CD A1 4B 7A : 07
6508 C3 A1 4B CD 0E 65 7E 23 : 90
6510 C3 A1 4B 1A B7 C8 13 BE : 19
6518 20 F9 37 C9 01 11 21 22 : 6E
6520 2A 31 32 3A C2 C3 C4 CA : DA
6528 CC CD D2 D4 DA DC E2 E4 : BB
6530 EA EC F2 F4 FA FC 00 10 : C2
6538 18 20 28 30 38 CB D3 DB : 41
6540 00 43 4B 53 5B 73 7B 00 : 2A
6548 11 A1 66 1A 4F 13 1A 47 : F5
6550 13 78 B1 C8 7C B8 20 02 : 5A
6558 7D B9 28 04 13 13 18 EB : 8B
6560 1A 4F 13 1A 47 03 7E 23 : 81
6568 CD A1 4B 7C B8 20 02 7D : 8C
6570 B9 20 F3 37 C9 3A A5 5D : 08
6578 F5 CD D3 39 F1 32 A5 5D : F3
---------------------------------
SUM: 4E BF F0 9C 53 25 0D A4 C36E

6580 C9 3A A5 5D F5 CD F5 52 : 0E
6588 18 F2 32 68 63 2A 0E 30 : 6F
6590 22 70 1F 44 4D 7E 23 B7 : 9A
6598 20 FB ED 42 22 72 1F C9 : C6
65A0 CD A1 4B 3E 00 32 67 63 : F3
65A8 C9 00 00 00 00 00 00 00 : C9
65B0 00 00 00 00 00 00 00 00 : 00
65B8 00 00 00 00 00 00 00 00 : 00
65C0 00 00 00 00 00 00 00 00 : 00
65C8 00 00 00 00 00 00 00 00 : 00
65D0 00 00 00 00 00 00 00 00 : 00
65D8 00 00 00 00 00 00 00 00 : 00
65E0 00 00 00 00 00 00 00 00 : 00
65E8 00 00 00 00 00 00 00 00 : 00
65F0 00 00 00 00 00 00 00 00 : 00
65F8 00 00 00 00 00 00 00 00 : 00
---------------------------------
SUM: B9 38 2E 89 C7 19 AC 65 C9C8

6600 00 00 00 00 00 00 00 00 : 00
6608 00 00 00 00 00 00 00 00 : 00
6610 00 00 00 00 00 00 00 00 : 00
6618 00 00 00 00 00 00 00 00 : 00
6620 00 00 00 00 00 00 00 00 : 00
6628 00 00 00 00 00 00 00 00 : 00
6630 00 00 00 00 00 00 00 00 : 00
6638 00 00 00 00 00 00 00 00 : 00
6640 00 00 00 00 00 00 00 00 : 00
6648 00 00 00 00 00 00 00 00 : 00
6650 00 00 00 00 00 00 00 00 : 00
6658 00 00 00 00 00 00 00 00 : 00
6660 00 00 00 00 00 00 00 00 : 00
6668 00 00 00 00 00 00 00 00 : 00
6670 00 00 00 00 00 00 00 00 : 00
6678 00 00 00 00 00 00 00 00 : 00
---------------------------------
SUM: 00 00 00 00 00 00 00 00 0000

6680 00 00 00 00 00 00 00 00 : 00
6688 00 00 00 00 00 00 00 00 : 00
6690 00 00 00 00 00 00 00 00 : 00
6698 00 00 00 00 00 00 00 00 : 00
66A0 00 2F 6C 30 6C 3D 6B 42 : 21
66A8 6B 90 6C 98 6C 80 6D A1 : F9
66B0 6D 00 00 00 00 00 00 00 : 6D
66B8 00 00 00 00 00 00 00 00 : 00
66C0 00 00 00 00 00 00 00 00 : 00
66C8 00 00 00 00 00 00 00 00 : 00
66D0 00 00 00 00 00 00 00 C0 : C0
66D8 19 00 4D 41 49 4E 0D 00 : 4B
66E0 41 00 00 46 41 4C 53 45 : AC
66E8 0D 41 01 00 54 52 55 45 : 8F
66F0 0D 44 90 53 50 52 49 4E : 6D
66F8 54 0D 44 E8 52 43 4F 44 : B5
---------------------------------
SUM: A0 51 FA 8A 58 3E 25 BF DDA9

6700 45 0D 44 A8 52 50 4F 4B : 7A
6708 45 0D 45 4A 42 4D 45 4D : 02
6710 0D 45 45 42 4D 45 4D 57 : 0F
6718 0D 45 D6 42 50 4F 52 54 : AF
6720 0D 45 DB 42 50 4F 52 54 : B4
6728 57 0D 45 32 43 53 4F 53 : 13
6730 0D 45 37 43 53 4F 53 57 : 18
6738 0D 45 77 43 43 4F 4E 53 : 3F
6740 54 0D 45 7C 43 43 4F 4E : 45
6748 53 54 57 0D 80 C4 1F 42 : B0
6750 45 45 50 0D 01 76 1F 40 : BD
6758 4B 42 55 46 46 0D 00 00 : 7B
6760 00 00 00 00 00 00 00 00 : 00
6768 00 00 00 00 00 00 00 00 : 00
6770 00 00 00 00 00 00 00 00 : 00
6778 00 00 00 00 00 00 00 00 : 00
---------------------------------
SUM: 59 68 B3 4C 64 FB 02 64 BCBA

6780 00 00 00 00 00 00 00 00 : 00
6788 00 00 00 00 00 00 00 00 : 00
6790 00 00 00 00 00 00 00 00 : 00
6798 00 00 00 00 00 00 00 00 : 00
67A0 00 00 00 00 00 00 00 00 : 00
67A8 00 00 00 00 00 00 00 80 : 80
67B0 29 00 53 54 4F 50 0D 81 : FD
67B8 01 30 02 41 42 53 0D 81 : 97
67C0 01 40 02 53 45 58 0D 81 : C1
67C8 01 36 02 53 47 4E 0D 81 : AF
67D0 01 84 02 52 4E 44 0D 81 : F9
67D8 02 47 02 42 49 54 0D 81 : B8
67E0 02 52 02 53 45 54 0D 81 : D0
67E8 02 62 02 52 45 53 45 54 : E9
67F0 0D 80 7C 03 49 4E 50 55 : 48
67F8 54 0D 81 01 26 03 49 4E : A3
---------------------------------
SUM: 94 B2 5E 78 AD D9 39 FE F261

6800 4B 45 59 0D 81 01 46 03 : C1
6808 47 45 54 4C 0D 81 02 48 : 04
6810 03 47 45 54 4C 49 4E 0D : D3
6818 81 02 3E 03 4C 49 4E 50 : F7
6820 55 54 0D 81 01 74 02 57 : 05
6828 49 44 54 48 0D 81 01 0B : C3
6830 03 50 52 4D 4F 44 45 0D : D7
6838 81 02 78 02 4C 4F 43 41 : 1C
6840 54 45 0D 81 02 7C 02 53 : FA
6848 43 52 45 45 4E 0D 81 02 : FD
6850 09 02 56 54 4F 53 0D 80 : E4
6858 CE 02 47 45 54 52 45 47 : 8E
6860 0D 81 01 A3 02 43 41 4C : 04
6868 4C 0D 01 AE 02 5E 42 43 : ED
6870 0D 01 B1 02 5E 44 45 0D : B5
6878 01 B4 02 5E 48 4C 0D 01 : B7
---------------------------------
SUM: 0D 9B FF D8 6C FB 19 11 6300

6880 B8 02 5E 49 58 0D 01 BC : 83
6888 02 5E 49 59 0D 01 03 03 : 16
6890 5E 41 0D 01 02 03 5E 41 : 51
6898 46 0D 01 05 03 5E 53 50 : 5D
68A0 0D 01 05 03 5E 53 54 41 : 5C
68A8 43 4B 0D 01 07 03 5E 43 : 47
68B0 41 52 52 59 0D 01 07 03 : 56
68B8 5E 43 59 0D 01 09 03 5E : 72
68C0 5A 45 52 4F 0D 81 03 14 : E5
68C8 04 46 4F 50 45 4E 0D 81 : 0A
68D0 03 11 05 46 53 45 45 4B : 87
68D8 0D 81 02 86 05 46 50 55 : 06
```

▶ミュージカル・プランのマジックパレットに描いてあるスイカの絵が，みょーに気にいってしまいました。それからというものの，スイカを食べたくてしょうがありません。
岩田　英之(13)茨城県

```
68E0 54 43 0D 81 01 65 06 46 : D7
68E8 47 45 54 43 0D 81 01 29 : DB
68F0 07 46 43 4C 4F 53 45 0D : D0
68F8 00 00 00 00 00 00 00 00 : 00
---------------------------------
SUM: 5D 7A BE 8D E4 62 62 E6 79EB

6900 00 00 00 00 00 00 00 00 : 00
6908 00 00 00 00 00 00 00 00 : 00
6910 00 00 00 00 00 00 00 00 : 00
6918 00 00 00 00 00 00 00 00 : 00
6920 00 00 00 00 00 00 00 00 : 00
6928 00 00 00 00 00 00 00 00 : 00
6930 00 00 00 00 00 00 00 00 : 00
6938 00 00 00 00 00 00 00 00 : 00
6940 00 00 00 00 00 00 00 00 : 00
6948 00 00 00 00 00 00 00 00 : 00
6950 00 00 00 00 00 00 00 00 : 00
6958 00 00 00 00 00 00 00 00 : 00
6960 00 00 00 00 00 00 00 00 : 00
6968 00 00 00 00 00 00 00 00 : 00
6970 00 00 00 00 00 00 00 00 : 00
6978 00 00 00 00 00 00 00 00 : 00
---------------------------------
SUM: 00 00 00 00 00 00 00 00 0000

6980 00 00 00 00 00 00 00 00 : 00
6988 00 00 00 00 00 00 D9 E1 : BA
6990 22 B5 69 D9 C5 D5 E5 DD : 75
6998 E5 FD E5 ED 73 AB 69 00 : 3B
69A0 00 00 FD 21 00 00 CD 00 : EB
69A8 00 B7 31 00 00 FD E1 DD : A3
69B0 E1 E1 D1 C1 C3 00 00 37 : 4E
69B8 18 F0 CD 94 1F 5F 16 00 : FD
69C0 C9 CD 94 1F 5F 23 CD 94 : 2C
69C8 1F 57 2B C9 7B C3 9A 1F : 61
69D0 7B CD 9A 1F 23 7A C3 9A : FB
69D8 1F E1 ED B0 E9 7D 93 7C : 12
69E0 9A 30 01 EB 44 4D 21 00 : 68
69E8 00 7A B7 28 0D 1F CB 1B : 6B
69F0 30 01 09 CB 21 CB 10 B7 : B8
69F8 20 F3 7B 1F 30 01 09 CB : B2
---------------------------------
SUM: 6C AA 9C F0 A2 F1 AD 38 60E2

6A00 21 CB 10 B7 20 F5 C9 7D : 0E
6A08 93 7C 9A 30 05 EB 21 00 : EA
6A10 00 C9 14 15 28 18 4D 6C : EB
6A18 AF 67 06 08 87 CB 21 ED : 84
6A20 6A ED 52 30 02 19 3D 3C : 6D
6A28 10 F2 EB 60 6F C9 1C 1D : BE
6A30 20 05 EB 21 00 00 C9 AF : A9
6A38 06 10 24 25 20 04 06 08 : 91
6A40 65 6F 29 8F BB 38 02 93 : 14
6A48 2C 10 F7 5F C9 7C AA 47 : C8
6A50 CB 7C C4 7B 6A EB CB 7C : 22
6A58 C4 7B 6A EB CB 78 28 A7 : A6
6A60 CD 07 6A 18 16 CD 07 6A : AA
6A68 EB C9 EB CB 7C C4 7B 6A : 8F
6A70 EB CB 7C 28 F0 CD 7B 6A : FC
6A78 CD 65 6A 2B 7C 2F 67 7D : 56
---------------------------------
SUM: 93 E1 99 64 1C 4D 83 9E 6563

6A80 2F 6F C9 7C B5 21 00 00 : B9
6A88 C0 2C C9 7C B5 C8 21 01 : D0
6A90 00 C9 EB B7 ED 52 21 00 : CB
6A98 00 D0 2C C9 EB B7 ED 52 : A6
6AA0 21 00 00 D8 2C C9 EB CB : A4
6AA8 7C 20 08 CB 7A 28 E4 21 : 16
6AB0 00 00 C9 CB 7A 20 DC 21 : 2B
6AB8 01 00 C9 EB CB 7C 20 08 : 24
6AC0 CB 7A 28 D9 21 01 00 C9 : 31
6AC8 CB 7A 20 D1 21 00 00 C9 : 20
6AD0 7B E6 0F C8 29 3D 20 FC : BA
6AD8 C9 7B E6 0F C8 CB 3C CB : D3
6AE0 1D 3D 20 F9 C9 7B E6 0F : AC
6AE8 C8 CB 2C CB 1D 3D 20 F9 : FD
6AF0 C9 7D B3 6F 7C B2 67 C9 : C6
6AF8 7D A3 6F 7C A2 67 C9 7D : 5A
---------------------------------
SUM: 92 D1 EE 01 64 59 8C 0F E6B8

6B00 AB 6F 7C AA 67 C9 1E 1C : AA
6B08 18 06 1E 20 18 02 1E 0D : A1
6B10 EB 7A B3 C8 7D CD 3A 6B : CF
```

```
6B18 1B 18 F6 7C CD 3A 6B 7D : 94
6B20 18 18 3E 0D 18 14 7C CD : F0
6B28 2B 6B 7D F5 0F 0F 0F 0F : 44
6B30 CD BB 1F CD 3A 6B F1 CD : D7
6B38 BB 1F C3 F4 1F 31 32 33 : 46
6B40 34 35 00 CB 7C 28 08 3E : 1E
6B48 2D CD 3A 6B CD 7B 6A 11 : 62
6B50 FF FF 18 03 11 05 00 D5 : 04
6B58 11 3D 6B CD 97 6B EB D1 : 44
6B60 7B FE 05 30 09 3E 05 93 : 8D
6B68 23 3D 20 FC 18 0D 7B FE : 1A
6B70 FF 20 08 7E FE 20 20 03 : E6
6B78 23 18 F8 06 00 7E B8 C8 : 37
---------------------------------
SUM: C5 15 C2 87 59 8D 44 3E 4E48

6B80 CD 3A 6B 23 18 F7 06 0D : B7
6B88 18 F3 E3 7E 23 B7 28 05 : 73
6B90 CD 3A 6B 18 F6 E3 C9 E5 : 11
6B98 D9 E1 D9 21 05 00 19 36 : 08
6BA0 00 06 05 D9 1E 0A CD 2E : 07
6BA8 6A 7B C6 30 D9 2B 77 10 : 66
6BB0 F2 06 04 7E FE 30 20 05 : CD
6BB8 36 20 23 10 F6 C9 CB 7C : 8F
6BC0 C4 7B 6A C9 7D B4 C8 17 : 82
6BC8 21 01 00 D0 18 F2 CB 7D : 44
6BD0 26 00 C8 25 C9 CD D9 6A : EC
6BD8 CB 45 21 00 00 C8 23 C9 : E5
6BE0 EB 7D E6 0F 21 01 00 28 : A7
6BE8 04 29 3D 20 FC C3 F1 6A : A4
6BF0 EB 7D E6 0F 21 FE FF 28 : A3
6BF8 06 37 ED 6A 3D 20 FA C3 : AE
---------------------------------
SUM: D3 0A CD D7 FA DC B8 30 C219

6C00 F8 6A 71 C3 30 20 63 C3 : 18
6C08 1E 20 63 CD 1B 20 6F 26 : 3E
6C10 00 C9 E5 2A 2F 6C 11 83 : 07
6C18 03 CD DD 69 7D 6C ED 5F : 4B
6C20 84 67 22 2F 6C D1 7A B3 : A6
6C28 20 02 EB C9 C3 65 6A 33 : 9B
6C30 E9 FD E5 11 4D 6C D5 E5 : 4F
6C38 3A 91 6C 01 00 00 11 00 : 49
6C40 00 21 00 00 DD 21 00 00 : 1F
6C48 FD 21 00 00 C9 E5 CD 5C : F5
6C50 6C 21 06 00 39 22 93 6C : ED
6C58 E1 FD E1 C9 E5 FD 22 4A : D6
6C60 6C DD 22 46 6C 22 42 6C : ED
6C68 ED 53 3F 6C ED 43 3C 6C : C3
6C70 F5 E1 22 90 6C 21 00 00 : 15
6C78 30 01 23 22 95 6C 21 00 : 98
---------------------------------
SUM: A8 89 8D 5A 91 D1 BB 80 BA69

6C80 00 20 01 23 22 97 6C 21 : 8A
6C88 04 00 39 22 93 6C E1 C9 : 08
6C90 00 00 00 00 00 00 00 00 : 00
6C98 00 7D FE 01 21 F4 1F 30 : E0
6CA0 05 CD D6 1F 18 0A 20 05 : 0E
6CA8 CD D9 1F 18 03 21 DC 1F : FC
6CB0 22 3B 6B C9 7D FE 01 30 : 3D
6CB8 05 CD D0 1F 18 0A 20 05 : 08
6CC0 CD 21 20 18 03 CD CA 1F : DF
6CC8 6F 26 00 C9 E5 CD 18 20 : 48
6CD0 55 E1 18 04 1E 00 16 00 : 86
6CD8 D5 ED 5B 76 1F CD D3 1F : 71
6CE0 C1 1A FE 1B 20 05 77 21 : B1
6CE8 FF FF C9 04 05 28 08 1A : 1A
6CF0 B7 28 04 13 05 18 F4 06 : 0D
6CF8 00 1A 13 B7 28 06 77 23 : AC
---------------------------------
SUM: DA BB D9 A9 FD DC 3E 35 BAA2

6D00 04 0D 20 F5 36 00 68 26 : EA
6D08 00 C9 01 00 00 ED 43 95 : 8F
6D10 6C 2A 76 1F 11 00 00 CD : 09
6D18 CC 6C ED 5B 76 1F 1A FE : 2D
6D20 20 20 03 13 18 F8 1A FE : 7E
6D28 24 20 07 13 CD B2 1F 38 : 34
6D30 1C C9 21 00 00 1A CD 58 : 45
6D38 6D 38 12 29 44 4D 29 29 : C3
6D40 09 06 00 4F 09 13 1A CD : 61
6D48 58 6D 30 EF C9 01 01 00 : AF
6D50 ED 43 95 6C 21 00 00 C9 : 1B
6D58 D6 30 D8 FE 0A 3F C9 00 : EE
6D60 FF FF 00 00 FF FF 00 00 : FC
```

```
6D68 FF FF 00 00 FF FF 00 00 : FC
6D70 FF FF 00 00 FF FF 00 00 : FC
6D78 FF FF 00 00 FF FF 00 00 : FC
---------------------------------
SUM: 29 8F 5E 66 DF 6C D8 D3 9A92

6D80 00 00 00 00 00 00 00 00 : 00
6D88 00 00 00 00 00 00 00 00 : 00
6D90 00 00 00 00 00 00 00 00 : 00
6D98 00 00 00 C8 00 C9 00 CA : 5B
6DA0 00 CB CD 06 71 FD 7E 09 : 93
6DA8 B7 C2 1D 71 E5 D5 CD 78 : 06
6DB0 6E D1 E1 69 60 7D FD 77 : DA
6DB8 00 FE 04 3E 0E 3F DA 78 : DF
6DC0 6E 3E 04 CD A3 1F DA 78 : 91
6DC8 6E CD 63 28 3E 0F CC 78 : 57
6DD0 6E CA FA 70 AF FD 77 03 : C8
6DD8 FD 77 04 FD 77 05 FD 7E : 6C
6DE0 00 FE 03 CA 20 6E CD 09 : 2F
6DE8 20 DA 78 6E 2A 74 1F FD : 9A
6DF0 7E 00 B7 28 07 7E CD 7C : 2B
6DF8 25 DA 78 6E 11 1E 00 19 : 2D
---------------------------------
SUM: 2F 5A DE 16 2D 05 F5 46 8C01

6E00 7E FD 77 01 FD 77 03 ED : 57
6E08 5B 5E 1F 2A 62 1F 3E 01 : C2
6E10 FD 77 09 CD 00 20 3A 5D : 01
6E18 1F FD 77 0A B7 C3 78 6E : FD
6E20 21 00 00 22 72 1F 22 6E : 64
6E28 1F 22 70 1F CD AF 1F DA : 45
6E30 78 6E CD 36 27 FD 77 01 : 85
6E38 FD 77 03 2A 74 1F 11 1D : 62
6E40 00 19 36 00 23 77 23 36 : 42
6E48 00 ED 5B E1 27 2A 74 1F : 0D
6E50 01 20 00 ED B0 ED 5B DF : E5
6E58 27 2A 64 1F 3E 01 FD 77 : 87
6E60 09 CD 03 20 FD 7E 01 2A : 9F
6E68 62 1F 85 30 01 24 6F 36 : 00
6E70 80 B7 3A 5D 1F FD 77 0A : 6B
6E78 2A 62 1F DD 5E 02 DD 56 : 1B
---------------------------------
SUM: E7 2B 2C 1A A3 93 6F 8A 68EF

6E80 03 ED 53 62 1F DD 75 02 : 18
6E88 DD 74 03 2A 64 1F DD 5E : 3C
6E90 00 DD 56 01 ED 53 64 1F : F7
6E98 DD 75 00 DD 74 01 C9 CD : 3A
6EA0 06 71 FD 7E 0A 32 5D 1F : AA
6EA8 CD FC 70 3E 01 B9 EB 3E : 5A
6EB0 0E D8 CA D3 6E 7D FD 77 : E2
6EB8 05 7C E6 0F FD 77 04 7C : 6A
6EC0 CB 3F CB 3F CB 3F CB 3F : 28
6EC8 FD 77 02 FD 75 0B FD 74 : 64
6ED0 0C B7 C9 E5 FD 5E 0B FD : D4
6ED8 56 0C 19 FD 75 0B FD 74 : 69
6EE0 0C E1 FD 7E 05 85 30 01 : 23
6EE8 24 F5 FD 77 05 FD 7E 04 : 11
6EF0 84 F5 E6 0F FD 77 04 F1 : D7
6EF8 1F 1F 1F 1F E6 1F 67 F1 : D9
---------------------------------
SUM: A0 D7 77 49 F9 FA B1 A7 D752

6F00 7C 20 02 F6 10 FD E5 E1 : 67
6F08 23 23 86 77 FD 6E 0B FD : B6
6F10 66 0C B7 C9 CD 06 71 FD : 33
6F18 7E 0A 32 5D 1F CD FC 70 : 6F
6F20 FD 7E 00 B7 3E 10 CA FA : 44
6F28 70 D5 FD 7E 02 B7 CA 54 : 97
6F30 6F 47 FD 7E 03 DD 6E 02 : 81
6F38 DD 66 03 85 6F 30 01 24 : 8F
6F40 7E FE 80 38 04 CD 21 71 : 97
6F48 77 FD 77 03 05 C2 35 6F : 59
6F50 AF FD 77 02 DD 6E 02 DD : 4F
6F58 66 03 FD 7E 03 85 6F 30 : 0B
6F60 01 24 46 AF B8 CA 6F 6F : 7A
6F68 3E 7F B8 D2 79 6F 04 FD : 30
6F70 7E 04 C6 80 B8 DA 79 6F : 42
6F78 77 FD 7E 03 6F 26 00 29 : B3
---------------------------------
SUM: 7A F8 1B 8A EC CD 13 B0 803B

6F80 29 29 29 FD 7E 04 85 6F : EE
6F88 30 01 24 EB FD 7E 06 FE : BF
6F90 02 C2 B6 6F FD 6E 07 FD : 58
6F98 66 08 B7 ED 52 C4 CF 6F : 66
```

▶とうとう英単語連想記憶術に3冊目がでました。これによるとCombine（結合する）は"「今晩いいん？」と結合する"と覚えるそうで……。Beastもちょっとあぶないけど，Createなどはとても恥ずかしい。
牧野 裕(18)北海道

```
6FA0 DD 6E 00 DD 66 01 FD 7E : 0A
6FA8 05 85 6F 30 01 24 C1 71 : 80
6FB0 21 01 00 C3 D3 6E 3E 02 : 66
6FB8 FD 77 06 FD 73 07 FD 72 : 60
6FC0 08 DD 6E 00 DD 66 01 3E : D5
6FC8 01 CD 00 20 C3 A0 6F D5 : 95
6FD0 FD 5E 07 FD 56 08 DD 6E : 08
6FD8 00 DD 66 01 3E 01 CD 03 : 53
6FE0 20 D1 FD 73 07 FD 72 08 : DF
6FE8 DD 6E 00 DD 66 01 3E 01 : CE
6FF0 C3 00 20 CD 06 71 FD 7E : A2
6FF8 0A 32 5D 1F CD FC 70 FD : EE
---------------------------------
SUM: 91 B5 84 6B EB C8 91 44 7E84

7000 7E 00 FE 01 3E 10 CA FA : 8F
7008 70 FD 7E 02 B7 CA 31 70 : 0F
7010 47 DD 6E 02 DD 66 03 FD : D7
7018 7E 03 85 6F 30 01 24 7E : 48
7020 FE 80 3F DA F8 70 FD 77 : 73
7028 03 05 C2 11 70 AF FD 77 : 6E
```

```
7030 02 DD 6E 02 DD 66 03 FD : 92
7038 7E 03 85 6F 30 01 24 AF : 79
7040 BE CA F8 70 3E 7F BE 30 : 9B
7048 09 FD 7E 04 C6 7F BE D2 : 5D
7050 F8 70 FD 7E 03 6F 26 00 : 7B
7058 29 29 29 29 FD 7E 04 85 : A8
7060 6F 30 01 24 EB FD 7E 06 : 30
7068 B7 CA 8F 70 FE 01 CA A8 : F1
7070 70 CD CF 6F DD 6E 00 DD : A3
7078 66 01 FD 7E 05 85 6F 30 : 0B
---------------------------------
SUM: 18 6A 5B 6C 46 A3 A0 C1 C18C

7080 01 24 E5 21 01 00 CD D3 : CC
7088 6E E1 7E 6F 26 00 C9 3E : 69
7090 01 FD 77 06 FD 73 07 FD : EF
7098 72 08 DD 6E 00 DD 66 01 : 09
70A0 3E 01 CD 00 20 C3 74 70 : D3
70A8 FD 6E 07 FD 66 08 B7 ED : 81
70B0 52 CA 74 70 C3 94 70 CD : 94
70B8 06 71 FD 7E 0A 32 5D 1F : AA
```

```
70C0 FD 7E 06 FE 02 CC E7 70 : A4
70C8 AF FD 77 09 32 1E 29 FD : A2
70D0 77 06 FD 7E 00 B7 C8 ED : 64
70D8 5B 5E 1F DD 6E 02 DD 66 : 68
70E0 03 3E 01 CD 03 20 C9 FD : F8
70E8 5E 07 FD 56 08 DD 6E 00 : 0B
70F0 DD 66 01 3E 01 C3 03 20 : 69
70F8 3E 11 37 C9 FD 7E 09 B7 : 8A
---------------------------------
SUM: 6F 4F CB 7B 22 C2 F3 EC F6E2

7100 C0 F1 37 3E 0C C9 7D B7 : 2F
7108 C2 14 71 DD 21 9A 6D FD : 49
7110 21 80 6D C9 DD 21 9E 6D : E0
7118 FD 21 8D 6D C9 3E 12 37 : 68
7120 C9 C5 E5 06 80 DD 6E 02 : 46
7128 DD 66 03 7E B7 28 06 23 : CC
7130 10 F9 37 18 04 3E 80 90 : AA
7138 B7 E1 C1 C9             : 22
---------------------------------
SUM: 0D AB 82 B6 0E 05 8E 0D 2AA9
```

全機種共通システムインデックス

■85年6月号
序論　共通化の試み
第1部　S-OS"MACE"
第2部　Lisp-85インタプリタ
第3部　チェックサムプログラム
■85年7月号
第4部　マシン語プログラム開発入門
第5部　エディタアセンブラZEDA
第6部　デバッグツールZAID
■85年8月号
第7部　ゲーム開発パッケージBEMS
第8部　ソースジェネレータZING
■85年9月号
インタラプト　S-OS番外地
第9部　マシン語入力ツールMACINTO-S
第10部　Lisp-85入門(1)
■85年10月号
第11部　仮想マシンCAP-X85
連載　Lisp-85入門(2)
■85年11月号
連載　Lisp-85入門(3)
■85年12月号
第12部　Prolog-85発表
■86年1月号
第13部　リロケータブルのお話
第14部　FM音源サウンドエディタ
■86年2月号
第15部　S-OS"SWORD"
第16部　Prolog-85入門(1)
■86年3月号
第17部　magiFORTH発表
連載　Prolog-85入門(2)
■86年4月号
第18部　思考ゲームJEWEL
第19部　LIFE GAME
連載　基礎からのmagiFORTH
連載　Prolog-85入門(3)
■86年5月号
第20部　スクリーンエディタE-MATE
連載　実戦演習magiFORTH
■86年6月号
第21部　Z80TRACER
第22部　magiFORTH TRACER
第23部　ディスクダンプ&エディタ
第24部　"SWORD" 2000 QD
連載　対話で学ぶ magiFORTH
特別付録　PC-8801版S-OS"SWORD"
■86年7月号
第25部　FM音源ミュージックシステム
付録　FM音源ボードの製作
連載　計算力アップのmagiFORTH
特別付録　SMC-777版 S-OS"SWORD"
■86年8月号
第26部　対局五目並べ
第27部　MZ-2500版 S-OS"SWORD"
■86年9月号
第28部　FuzzyBASIC 発表
連載　明日に向かって magiFORTH
■86年10月号
第29部　ちょっと便利な拡張プログラム
第30部　ディスクモニタ DREAM
第31部　FuzzyBASIC 料理法〈1〉
■86年11月号
第32部　パズルゲーム HOTTAN
第33部　MAZE in MAZE
連載　FuzzyBASIC 料理法〈2〉
■86年12月号
第34部　CASL & COMET
連載　FuzzyBASIC 料理法〈3〉
■87年1月号
第35部　マシン語入力ツールMACINTO-C
連載　FuzzyBASIC 料理法〈4〉
■87年2月号
第36部　アドベンチャーゲーム MARMALADE
第37部　テキアベ作成ツール CONTEX
■87年3月号
第38部　魔法使いはアニメがお好き
第39部　アニメーションツール MAGE
付録　"SWORD" 再掲載と MAGIC の標準化
■87年4月号
第40部　INVADER GAME
第41部　TANGERINE
■87年5月号
第42部　S-OS"SWORD" 変身セット
第43部　MZ-700用 "SWORD" を QD 対応に
■87年6月号
インタラプト　コンパイラ物語
第44部　FuzzyBASIC コンパイラ
第45部　エディタアセンブラ ZEDA-3
■87年7月号
第46部　STORY MASTER
■87年8月号
第47部　パズルゲーム碁石拾い
第48部　漢字出力パッケージ JACKWRITE
特別付録　FM-7/77版 S-OS"SWORD"
■87年9月号
第49部　リロケータブル逆アセンブラ Inside-R
特別付録　PC-8001/8801 版 S-OS"SWORD"
■87年10月号
第50部　tiny CORE WARS
第51部　FuzzyBASIC コンパイラの拡張
第52部　X1turbo 版 S-OS"SWORD"
■87年11月号
序論　神話のなかのマイクロコンピュータ
付録　S-OS の仲間たち
第53部　もうひとつの FuzzyBASIC 入門
第54部　ファイルアロケータ&ローダ
インタラプト　S-OS こちら集中治療室
第55部　BACK GAMMON
■87年12月号
第56部　タートルグラフィックパッケージTURTLE
第57部　X1turbo 版 "SWORD" アフターケア ラインプリントルーチン
特別付録　PASOPIA7 版 S-OS"SWORD"
■88年1月号
第58部　FuzzyBASIC コンパイラ・奥村版
付録　石上版コンパイラ拡張部の修正
■88年2月号
第59部　シューティングゲーム ELFES
■88年3月号
第60部　構造型コンパイラ言語 SLANG
■88年4月号
第61部　デバッギングツール TRADE
第62部　シミュレーションウォーゲーム WALRUS
■88年5月号
第63部　シューティングゲーム ELFES II
第64部　地底最大の作戦
■88年6月号
第65部　構造化言語 SLANG 入門(1)
第66部　Lisp-85 用 NAMPA シミュレーション
■88年7月号
第67部　マルチウィンドウドライバ MW-1
連載　構造化言語 SLANG 入門(2)
■88年8月号
第68部　マルチウィンドウエディタ WINER
■88年9月号
第69部　超小型エディタ TED-750
第70部　アフターケア WINER の拡張
■88年10月号
第71部　SLANG 用ファイル入出力ライブラリ
第72部　シューティングゲーム MANKAI
■88年11月号
第73部　シューティングゲーム ELFES Ⅳ
■88年12月号
第74部　ソースジェネレータ SOURCERY
■89年1月号
第75部　パズルゲーム LAST ONE
第76部　ブロックゲーム FLICK
■89年2月号
第77部　高速エディタアセンブラ REDA
特別付録　X1版 S-OS"SWORD"〈再掲載〉
■89年3月号
第78部　Z80用浮動小数点演算パッケージSOROBAN
■89年4月号
第79部　SLANG 用実数演算ライブラリ
■89年5月号
第80部　ソースジェネレータ RING
■89年6月号
第81部　超小型コンパイラTTC
■89年7月号
第82部　TTC用パズルゲーム　TICBAN
■89年8月号
第83部　CP/M用ファイルコンバータ
■89年9月号
第84部　生物進化シミュレーションBUGS
■89年10月号
第85部　小型インタプリタ言語TTI
■89年11月号
第86部　TTI用パズルゲーム PUSH BON!
■89年12月号
第87部　SLANG用リダイレクションライブラリ DIO. LIB

*以上のアプリケーションは，基本システムである S-OS "MACE" または S-OS"SWORD" がないと動作しませんのでご注意ください。

BACK ISSUES バックナンバー案内

ここには1989年1月号から12月号までをご紹介しました。現在1987年4，1988年5,6,9,1989年4～12月号までの在庫がございます。バックナンバーおよび定期購読のお申し込み方法については，174ページを参照してください。

1989

1月号(品切れ)
特集　いきなり初春からハードウェア
デジタル回路入門/電子サイコロ/乱数発生器/X1turbo バンクメモリ拡張/X68000用CP/M-80システム　他
1988年度GAME OF THE YEAR ノミネート作品発表
- MZ-2500用　Hyper Game Book
- LIVE in '89 エンデューロレーサー/アルルの女
- ようこそ，セガ・メガドライブ!!

C言語講座PRO-68K/Z80マシン語ゲーム工房
全機種共通システム　パズルゲーム LAST ONE/FLICK

2月号(品切れ)
特集　マシン語"でじたるざんまい"
アーキテクチャからのマシン語入門/アセンブラへの招待/超入門Z80マシン語活用術/X68000料理教室
THE SOFTOUCH 彩CRONE/Final Ver.3.2　他
- X1/X1turbo用RPG FLAME

Z80マシン語ゲーム工房　最終回　爆発，そして完成へ
C調言語講座PRO-68K (8) とおりゃんせなのである
OS-9/X68000入門(3)　ついに発売！OS-9/X68000
全機種共通システム　高速エディタアセンブラREDA

3月号(品切れ)
特集　BASIC"おもちゃ箱"
ピコピコゲームから重力シミュレーションまで
- X1/X1turboでMZ-700用スペハリ/ロボットゲームTAMA
- 数値演算を高速化　FLOAT2＋.X

OS-9/X68000入門(4) C言語の概要を見る
C調言語講座PRO-68K(9) ニホン語，不得意
新連載予告編X68000マシン語プログラミング入門
全機種共通システム 浮動小数点演算パッケージSOROBAN
THE SOFTOUCH/LIVE in'89/知能機械概論/猫とコンピュータ

4月号
特集　ゲーマーたちの"新深夜族"宣言
1988年度GAME OF THE YEAR
新連載 X68000マシン語プログラミング
- X1/X1turbo用パズルゲーム　ロボット衛兵
- MZ-700用ゲームパッケージ System-7B
- LIVE グラディウスII/ザ・スキーム/パワードリフト

連載　C調言語講座PRO-68K/OS-9/X68000入門
全機種共通システム　SLANG用実数演算ライブラリ
特別付録　X68000イメージCGポスター

5月号
特集　MIDIサウンドデータ料理術
LA音源をFM音源でシミュレート/X-BASICでMIDI制御
特別企画　第4回「言わせてくれなくちゃだワ」
- シャープパソコンフォーラム'89 in赤坂
- 詳解Human68k ver.2.0
- MZ-2500，X1/X1turbo用　戦略的ライトサイクルゲーム

連載　C調言語講座PRO-68K/ OS-9/X68000入門
X68000マシン語プログラミング
全機種共通システム　ソースジェネレータRING

6月号
特集　これからのXfamily
X68000に光磁気ディスクを/学習リモコンの製作
THE SOFTOUCH ライトニングバッカス/Might and MagicII他
- OPMA用外部関数による　KENBAN.BAS
- X1/X1turbo用ドライブゲーム　Spirit of Rally
- X1turboZ用　これ，パズルなんですか。

MZ-2500 MIDI入門(1)MIDIボードを作る
C調言語講座PRO-68K/X68000マシン語プログラミング
全機種共通システム　超小型コンパイラTTC

7月号
特集 3Dグラフィックへの飛翔
Zバッファアルゴリズム/スムースシェイディング 他
THE SOFTOUCH Terazzo PRO-68K/アドヴァンスト・ファンタジアン
新連載
DōGA・CGアニメーション講座
MZ-2500用グラフィックエディタ作成講座
マシン語カクテル in Z80's Bar
X-BASICプログラミング調理実習
全機種共通システム TTC用パズルゲームTIC BAN
X68000マシン語プログラミング/C調言語講座PRO-68K 他

8月号
特集1　X1プログラミングガイドブック
PCGの基礎から奥義まで/超高速ラインルーチン 他
特集2　3Dグラフィックの深淵へ
スキャンラインZバッファ/3Dモデリング 他
新連載 (で)のショートプロぱーてぃ
X68000マシン語プログラミング/C調言語講座 PRO-68K
X-BASICプログラミング調理実習/DōGA・CGA講座
MZ-2500用グラフィックエディタ/Z80's Bar 他
全機種共通システム CP/M用ファイルコンバータ

9月号
特集　活用ハードディスク&プリンタ
各社ハードディスク接続総チェック/ハードディスク雑学講座/COPYキーメニュー/ビデオプリンタ活用プログラム　他
THE SOFTOUCH ジェノサイド/琉球/mFORTH Compiler
- サイバースティックで遊ぶ　不思議な環境ソフトの世界
- X1/X1turbo用シューティングゲーム　Defeat X

Z80's Bar/MZ-2500グラフィックエディタ　他
[X68000] X-BASIC/マシン語/C調言語講座/DōGA・CGA
全機種共通システム　生物進化シミュレーションBUGS

10月号
特集　ゲーム面白心理学
ソーサリアン・宇宙からの訪問者/ファンタジーゾーンねじ式/ガウディ・バルセロナの風/サバッシュ　他
- MZ-700用シューティングゲームSide Roll-F
- X1/X1turdo用カードゲームBonding

ショートプロ/Z80's Bar/MZ-2500グラフィックエディタ
X68000マシン語/X-BASIC/C調言語講座/DōGA・CGA
THE SOFTOUCH Z'sTRIPHONY DIGITAL CRAFT/James68K
全機種共通システム　小型インタプリタ言語TTI

11月号
特集　microComputer入門
初歩からのCPU物語/RISCプロセッサの設計と製作
X68000&X1で周辺LSIを使いこなそう
連載
ショートプロ/Z80's Bar/MZ-2500グラフィックエディタ
X68000マシン語/X-BASIC/C調言語講座/DōGA・CGA
- X68000用カードゲームばばぬき

LIVE in '89　メタルホーク/オブ・ラ・ディ，オブ・ラ・ダ
THE SOFTOUCH　Stationery PRO-68K/リングマスター1
全機種共通システム　TTI用パズルゲームPUSH BON!

12月号
特集　Cプログラミングへの招待
付録　C言語簡易リファレンス
連載
ショートプロぱーてぃ/Z80's Bar
X68000マシン語/X-BASIC/DōGA・CGA
- Oh!X2周年特別企画「素粒子の声が聞こえる」
- X1/turbo用アクションゲームACTIVE UNIT

LIVE in '89　天空の城ラピュタ/ギャラクシーフォース
THE SOFTOUCH　38万キロの虚空/た～みのる2
全機種共通システム　SLANG用リダイレクションライブラリ

愛読者プレゼント

プレゼントの応募方法

とじ込みのアンケートはがきの該当項目をすべてご記入のうえ，希望するプレゼント番号をはがき右下のスペースにひとつ記入してお申し込みください。締め切りは1990年1月18日の到着分までとします。当選の発表は1990年3月号で行います。

1 サイクロンExpress

アンス・コンサルタンツ ☎092(522)6347

X68000用 5"2HD版 78,000円

1名

初心者でも簡単にレイトレースの画面を作成できるグラフィックツール。これで次回のコンテストを目指そう。

2 Misty 2

データウエスト ☎06(968)1236

X68000用 5"2HD版

5,000円

2名

モノクロの画面が臨場感をかもし出す「Misty」シリーズの第2弾。今回は，シナリオ5本を収録。

3 ヒーロー・オブ・ランス

ポニーキャニオン ☎03(221)3151

X1turbo用 5"2D版4枚組

7,500円

3名

「アドヴァンスト・ダンジョンズ・アンド・ドラゴンズ」の「ドラゴンランス戦記」が原作のアクションゲーム。

4 レナムのポスター

ヘルツ ☎03(371)3801

5名

マウス操作がうれしいAVRPG「レナム」の宣伝用ポスター。ほかではちょっと手に入らないもの。5名の方に。

5 カレンダー

5名

日本ソフトバンクの1990年度のカレンダー。コンパクトな卓上サイズ。月ごとにかわいいイラストが描かれている。

11月号プレゼント当選者

1リングマスターⅠ（大阪府）野瀬茂樹（石川県）石野吉紘（鳥取県）森下寛和 2ローグ・アライアンス（東京都）佐藤貴之（大阪府）光井浄二（広島県）猪原忠彦 3ソングファイル68Kシリーズ A.リゾーム症候群（長野県）蟹沢克仁（大阪府）黒岩智教 B.スケッチ（神奈川県）山梨毅（富山県）藤島充 4サイクロン オリジナルTシャツ（東京都）栗原健次 辻文次（静岡県）小杉文武（福井県）藪吉登（兵庫県）滝沢与一 5コーヒーキャンディ（広島県）安達宏治 （敬称略）

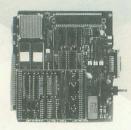
以上の方々が当選されました。おめでとうございます。商品は順次発送いたしますが，入荷状況などにより遅れる場合もあります。また，公正取引委員会の告示により，このプレゼントに当選された方は，この号の他の懸賞には当選できない場合がありますので，ご了承ください。

（価格はすべて消費税別です）

PENGUIN INFORMATION CORNER

ペ・ン・ギ・ン・情・報・コ・ー・ナ・ー

NEW PRODUCTS

X68000用LANボード
CZ-6BL1
シャープ

シャープは，X68000用LANボード「CZ-6BL1」の販売を開始した。

CZ-6BL1は，TCP/IPプロトコルをサポートするEthernetにX68000を接続するための拡張ボード。これにより，スタンドアロン環境のX68000をネットワークに統合し，データやプリンタの共用が可能になる。また，EWSやミニコンの端末として使用することも可能。使用できるコマンドは，telnet(リモートログイン)，ftp(ファイル転送)の2つ。価格は，268,000円となっている。また同時に，EthernetとCheapernetの両方に使用できる「CZ-6BL2」も発売され，こちらは298,000円。

〈問い合わせ先〉
シャープ(株)　☎06(621)1221，03(260)1161

カラー印刷可能な書院
WD-A900/A710
シャープ

シャープは，パーソナルワープロであるミニ書院シリーズの新製品としてカラー印刷可能な「WD-A900」(198,000円)と3.5インチフロッピー2基搭載の「WD-A710」(198,000円)の販売を開始する。両製品とも，新設計のAI-V2辞書を搭載している。

WD-A900は，業界初のトリプルカセット方式カラープリンタにより，印刷速度3倍・ランニングコスト1/5(同社比)を実現した。

また，カラー写真やイラストなどの入力ができるハンディカラースキャナを標準装備，オプションでビデオなどの画像を取り込むカラービデオアダプタも接続可能。

WD-A710は，毎秒50字で印刷できる56ドットの高速・高品位プリンタを搭載するとともに，3.5インチフロッピーディスクドライブを2基搭載したもの。

〈問い合わせ先〉
シャープ(株)　☎06(621)1221，03(260)1161

デザインシミュレーションシステム
CS-1
ユニテックジャパン

ユニテックジャパンは，X68000ベースのカラーデザインシミュレーションシステム「スーパーカラーシミュレーターCS-1」の販売を開始した。

システムは，40Mバイトハードディスク内蔵のX68000PROにカラーイメージスキャナ，カラービデオプリンタなどを組み合わせたもの。スキャナから読み込んだり，付属のペイントソフトで描いた画像を編集し，A6またはA4サイズでビデオプリンタに出力する。イメージ処理，マスク処理，パーツ処理などの編集作業は，すべて65 536色で行うことができる。ディスクには約40枚の画像を記録することができる。システム価格は3,500,000円と，従来のデザインシミュレーションシステムに比べると低価格である。

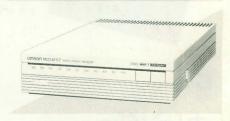
〈問い合わせ先〉
(株)ユニテックジャパン　☎06(946)7689

低価格MNP5モデム
通信ポコ
インテグラン

インテグランは，MNPクラス5の全二重モデム「通信ポコ(PoCo)」の販売を開始した。

同製品は，300～2400bpsの転送速度に対応したモデムで，エラー訂正方式にMNPクラス5を採用した。圧縮により最大4800bpsまで対応可能で，端末とモデム間の速度は最高19200bpsまで対応している。通信制御手順はヘイズAT互換で，価格は45,000円と市販のMNPクラス5モデムのなかでは最も低価格なクラスとなっている。

〈問い合わせ先〉
インテグラン(株)　☎03(546)1234

モデム新製品
MD24/96シリーズ
立石電機

立石電機は，CCITT V.42搭載の全二重モデムの販売を開始した。

販売開始されたのは，MNPクラス4搭載の「MD24FS4」(44,800円)，MNPクラス5の「MD24FS5」(49,800円)，MNPクラス7の「MD24FS7」(64,800円)の2400bpsのモデム3製品と，MNPクラス5で9600bpsの「MD96FS5」(198,000円)。各製品ともCCITT V.42をサポートし，MD96FS5はCCITT V.

32もサポートしている。制御手順は，ヘイズATおよびCCITT V.25bisに準拠。特に，MD24FS7は，国内で初めてMNPクラス7に対応したモデムで，実効通信速度は7200bps。全製品ともRS-232Cケーブルは付属している。

〈問い合わせ先〉
立石電機㈱　☎03(436)7266, 06(282)2672

プリンタバッファ PRBシリーズ
アイ・オー・データ機器

アイ・オー・データ機器は，2Mバイトと4Mバイトのプリンタバッファ「PRBシリーズ」の販売を開始する。同社ではすでに，500Kバイトタイプと1Mバイトタイプのプリンタバッファを販売しており，今回の製品で4機種が揃う。同シリーズは，機能面をシンプルにして他社に比べて低価格を図ったもので，価格は2Mバイトタイプが79,800円，4Mバイトタイプは未定。

〈問い合わせ先〉
アイ・オー・データ機器　☎0762(21)2373

ファミコン通信アダプター
マイクロコア

マイクロコアは，ファミコン通信，キャプテン端末機能，パソコン通信機能を持ったアダプタの販売を開始する。

ファミリーコンピュータかツインファミコンに装着して使用する通信アダプタで，キャプテン端末ランク2（ビデオテックス網），全二重 1200bps のパソコン通信（電話回線），ファミコン通信(電話回線，DDX-TP，VAN)の3つを1台で兼ねる。それぞれ4～12色のカラー表示ができ見やすいものとなっている。発売予定は90年4月で，価格は未定。

〈問い合わせ先〉
㈱マイクロコア　☎03(448)0811

時計付き電子メモ PA-220
シャープ

シャープは，電話番号やスケジュールなどを記憶する電子メモシリーズの新製品「PA-220」の販売を開始した。

PA-220は，電話番号なら450人分（名前カナで8文字，電話番号12桁），スケジュールなら540件分が記憶可能。また，スケジュールメモとして使った場合，予定が入っている日時になればアラームが鳴る。価格は，7,500円となっている。

〈問い合わせ先〉
シャープ㈱　☎06(621)1221, 03(260)1161

INFORMATION

OS-9000発表
マイクロウェア・システムズ

マイクロウェアはOS-9と上位互換性があるマルチユーザー，マルチタスクの32ビット用OSであるOS-9000を発表した。

OS-9000は，モトローラの680X0，インテルの80X86の両アーキテクチャのチップでの同一の操作環境を目指したもの。動作するCPUは68020, 68030, 80386などの32ビットチップとRISCチップ。各チップのOS-9000はCのソースコードレベルでは完全互換である。また，OSの移植性を高めるために，OS-9000カーネルの95%とI/Oモジュールやユーティリティなどの周辺ソフトはC言語で記述されている。さらに，TCP/IPやSLDCなどのプロトコル以外にISDNなどへのインタフェイスも備えている。

〈問い合わせ先〉
マイクロウェア・システムズ㈱
☎03(839)9000

BOOKS

3次元グラフィックス
アーマット

パソコン用CGソフトであるZ's Triphony(DIGITAL CRAFT)を利用した3次元グラフィックスの作り方を解説した本（玉川美土里著　2,500円）

まず，DIGITAL CRAFT 自身の使い方から始まり，立体のインビトウィーン，フラクタル図形の出力，モデリング過程詳細などを紹介している。CG用語集も付いているので初心者にもわかりやすいものとなっている。また，X68000の隠し機能全公開という項目もあるのでX68000ユーザーは一読を。

〈問い合わせ先〉
㈲アーマット　☎045(902)9041

低価格拡張スロットのお知らせ
システムショップハドソン

システムショップハドソンから，X68000用の拡張ボックスが発売される予定である。予定されているのは，4スロット内蔵（うち1スロットは本体接続用）のもので価格は3万円台になる見込み。外観デザイン（塗装は黒およびグレー）のコストを抑え，シャープ純正の拡張ボックスの88,000円に比べて大幅な低価格を実現した。製品の基本機能は純正品とほぼ同等である。同社では，ユーザーからの反響をみたうえで積極的な販売をするかどうかを判断するので，購入を希望するユーザーや興味があるユーザーは，同社まで問い合わせをしていただきたい。

〈問い合わせ先〉
システムショップハドソン　☎011(841)5155

FILES Oh!X

このインデックスは，タイトル，注記——筆者名，誌名，月号，ページで構成されています。めっきり寒くなってきました。受験生の方々はここがふんばりどきですね。頑張って良い春を迎えてください。

一般

▶特集　ネットマルチ接近遭遇
パソコン通信ネットで広まりつつある，マルチプレイヤーゲームについて特集。その歴史，発展経過，未来など。なお現在試験運用を始めた富士通ハビタットを紹介している。——編集部，LOGIN，21号，120-129pp.

▶図鑑世界のコンピュータちゃん　第23回
YHPとの合併で話題になったアポロ社のワークステーション，「DOMAINシリーズ」を紹介。ネットワークについて解説している。その他コンピュータ用語独断的解説大事典。——編集部，LOGIN，21号，160-161pp.

▶ネットワーカー・ホリック第10回
日本の大手有料ネットのひとつであるNIFTY-Serveを紹介。その特色としてフォーラムの充実を挙げている。その他，電子手帳を端末に使ったVANサービスを，シャープが来年半ばごろに始めるという話題など。——編集部，LOGIN，21号，218-221pp.

▶ネットワーカー・ホリック第11回
落語SIGのシグオペを務める落語家，三遊亭圓窓師匠を紹介。新開局ネットのDEMONはデモやサンプルソフトを提供するネット。——編集部，LOGIN，22号，234-237pp.

▶最新主力機種レポート'89冬
各社の新製品9種をレポート。98シリーズ全製品マップ付き。——編集部，ASCII，12月号，281-296pp.

▶NeXTディベロッパーズキャンプ
開発者を対象とした4日間の講習合宿の内容をレポート。——脇山弘敏，ASCII，12月号，319-325pp.

▶'89をしめくくる大型ショウにパソコン最前線動向を探る
データショウ，エレクトロニクスショウの各社ブースと製品のレポート。——編集部，マイコン，12月号，141-168pp.

▶ビジネスマンの情報管理術
電話帳と住所録機能の使い方の解説のあとは，プリンタを使った年賀状印刷にチャレンジだ。——塚田洋一，マイコン，12月号，324-329pp.

▶カッコウのたまご第1部
軍のコンピュータに侵入したネットワークハッカーと，それを迎え撃つ天文学者との対決をレポートしたノンフィクション。——Clifford Stoll著　椋田 直子訳，ASCII，12月号，329-336pp.

MZ-80K/C/1200/700/1500

MZ-700/1500（S-BASIC）

▶ナンテコッタイ99
UNOの姉妹品カードゲームO'NO99のパソコン版。——まっぴ，マイコンBASIC Magazine，12月号，128-130pp.

MZ-700/1500（HuBASIC）

▶雪ん子UFO
空から雪ん子が降ってくる！　しかしその雪ん子は，UFOだったのだ。地上に降りる前にビーム砲で阻止せよ。——林星児，マイコンBASIC Magazine，12月号，131p.

MZ-1500

▶PUP
パネルの上でかわいいキャラがとび跳ねて，敵を落とす。全15面。——吉田弘二，マイコンBASIC Magazine，12月号，132-134pp.

MZ-80B/2000/2500/2800

MZ-2200/2500（1Z001/1Z002）

▶MOUSE
ネズミの嫌いな主人公が，思わず落ちてしまったマンホールには，巨大なネズミがいたのです。迷路ゲーム。——トシちゃん27歳，マイコンBASIC Magazine，12月号，135-137pp.

MZ-2500

▶誌上公開質問状
MZ-2500の音源機能を上げるための拡張ボードは？——編集部/PEGUSUS，マイコンBASIC Magazine，12月号，64-65pp.

MZ-2500（BASIC-M25）

▶星のかけら
星のかけらを6個集める，奥の深いアクションゲーム。——蒲生敬，マイコンBASIC Magazine，12月号，138-140pp.

▶ザ・スキーム　～時の流れるままに～
ゲームミュージックプログラム。——RYO-3，マイコンBASIC Magazine，12月号，191-192pp.

X1/turbo/Z

X1シリーズ

▶誌上公開質問状
X1のI/Oポートについて詳しく掲載している本を紹介。——多田太郎，マイコンBASIC Magazine，12月号，65p.

▶FINISH
体操の床運動のフィニッシュをゲームにした。空中で宙返りをして，その出来栄えを競う。要ジョイスティック。——HARU，マイコンBASIC Magazine，12月号，166-167pp.

▶P-700
スポーツカーレースで現在最下位のあなた。5つのチェックポイントを通過するまでにトップを取ってくださ

参考文献
I/O　工学社
ASCII　アスキー
テクノポリス　徳間書店
POPCOM　小学館
マイコン　電波新聞社
マイコンBASIC Magazine　電波新聞社
LOGIN　アスキー

新刊書案内

ハッカー英語辞典
the HACKER'S DICTIONARY

アメリカの大学などにいるハッカーたちが協力して維持しているjargon fileというスラングの辞書がある。本書は，その改訂版の訳である。ハッカーといっても，ネットワークへの侵入やウイルスといった（善良な市民の期待する）イメージはどこにもない。HACKERの項も"1.コンピュータ・システムの細部や，その能力の伸ばし方を学ぶことを楽しむ人"だ。そこにあるのは，本当のコンピュータマニアが日常生活で使っているコンピュータ用語の束であり，ユーモアやウィットに富んだ彼らの生態なのである。

MITなど大学周辺の話が中心だから，いまいちわからない言葉が多いかもしれない。参考までに，話によく出るのはDEC-PDP-LISP-EMACSというラインだ。ハッカーはIBMよりDECを，CよりLISPを好むのだろう。それがYESの代わりにTという（もちろん，NOの代わりはNILだ）などに表れている。とはいえ，辞典としてでなく読み物として十分に面白いし，我々も日常会話でコンピュータ用語をかなり使っていることに気づかされて楽しい。（K）

ハッカー英語辞典　ガイ・L・スティル・ジュニア他5名著　犬伏茂之訳　自然社刊　産学社発売
☎03(261)3393(産学社) A5判 149ページ 1,400円

書籍の価格は消費税込みです

い。高速カーレースゲーム。——花光悟，マイコン BASIC Magazine，12月号，168-170pp.

▶ SL-MAZE

レールの敷かれた迷路を SL で脱出する。ライトの先は線路がどうなっているかわからないから，なかなか難しいぞ。——安永誠治，マイコン，12月号，230-236pp.

▶ PC→X1ファイルコンバータ

88，98の BASIC 標準フォーマットから X1へファイル転送。漢字コードと MML データの変換も OK。——真田智之，I/O，12月号，238-239pp.

X1＋FM 音源ボード（要 NEW FM 音源ドライバ）

▶オーダイン　エンディング BGM

ゲームミュージックプログラム。——上田順一，マイコン BASIC Magazine，12月号，197-198pp.

X1turbo シリーズ

▶最新ゲーム徹底解剖!!

ヒーロー・オブ・ランスのキャラクター，マップなどを紹介。——編集部，LOGIN，21号，194-197pp.

▶やっぱり大好きファンタジー RPG90's

ヒーロー・オブ・ランスを紹介。——編集部，POPCOM，12月号，56-57pp.

▶ラッカー01　——ガッタイダ——

悪の軍団を倒すために地球防衛隊が始動したラッカー01(?)。上から落ちてくる歯を下の歯と噛み合わせるゲーム。——大原コンピュータ同好会，マイコン BASIC Magazine，12月号，171-172pp.

▶なんでもQ＆A

X1turbo と家庭用テレビでスーパーインポーズはできるのか？　など。——編集部，マイコン，12月号，408-409pp.

X68000

▶ NEW SOFT

発売予定のゲームソフト，アークスIIと夢幻戦士ヴァリスII，新作アクションゲーム WINGS を紹介。——編集部，LOGIN，21号，14-17pp.

▶ X68000新聞

'89年度中に出荷台数10万台を越えようとしている X68000。その歴史，コンセプトを軽く解説。新着ゲームは，ナイトアームズ，ニュージーランドストーリー，メタルサイト，Misty Vol.1，た〜みのる2，開発中のサンダーブレード，スーパーハングオンを紹介。PDS は6〜8万色画像コンバータ GTOT を紹介。——編集部，LOGIN，21号，144-149pp.

▶ NEW SOFT

開発中のゲーム，夢幻戦士ヴァリスIIとフラッピー2を紹介している。——編集部，LOGIN，22号，25-26pp.

▶ X68000新聞

新着ゲーム，V'BALL，ラグーン，やじうまペナントレース，ゲーム開発用グラフィックツールのGツール，サイクロンCGコンテストの模様を紹介。PDSは暇つぶしソフトの「プチプチPRO68K」。——編集部，LOGIN，22号，162-167pp.

▶ SOFTWARE REVIEW

夢あふれるファンタジーアクションゲーム，ニュージーランドストーリーを紹介。——くしだナム子，LOGIN，22号，228-229pp.

▶ GAMING WORLD

開発中のゲーム，ディオス，ラグーン，A-JAX，メタルサイト，V'BALL，レナム，GAMMA PLANET を紹介。——編集部，テクノポリス，12月号，25-37pp.

▶攻略おすすめゲーム

発売予定のアークスIIの攻略法を他機種を参考にして解説。——編集部，テクノポリス，12月号，60-61pp.

▶最新ゲーム徹底紹介，ゲームがオレを呼んでいる！

戦国 SLG「斬［ZAN］陽炎の時代」を紹介。——稲生達朗，POPCOM，12月号，74-77pp.

▶ WE ARE THE X68000 WORLD

新着ゲーム，ナイトアームズ，A-JAX，GAMMA PLANET，レナム，バトルチェス，夢幻戦士ヴァリスII，ニュージーランドストーリーの隠しコマンドなど。——編集部，POPCOM，12月号，88-91pp.

▶誌上公開質問状

CZ-8PC2を使ってカラー絵はがきなどを作れるソフトや，CZ-8PC4を使って48ドット印字可能なワープロソフトを紹介。カセットプレイヤーからサンプリングをする方法を解説。——多田太郎，マイコン BASIC Magazine，12月号，65p.

▶マリモころがし GAME

2人で協力して遊ぶちょっと変わった旗取りゲーム。マリモを転がすために台のほうを動かすという意表を突いた面白さ。——高橋潤，マイコン BASIC Magazine，12月号，173-174pp.

▶ JUMP！ The QUEUE

2重スクロールのドライブアクションゲーム。——高橋秀之，マイコン BASIC Magazine，12月号，175-176pp.

▶ PLUS ALPHA 1面 BGM

ゲームミュージックプログラム。——石田勇，マイコン BASIC Magazine，12月号，199-201pp.

▶ PASOCOM SOFT

新作アクションゲームの WINGS を紹介。——佐久間亮介，マイコン BASIC Magazine，12月号，272-274pp.

▶チャレンジ！X68000

ビーチバレーのゲーム「V'BALL」と，近未来を舞台にした3D アクションゲーム「メタルサイト」，コナミのアーケードゲーム「A-JAX」の X68000版を紹介。——佐久間亮介，マイコン BASIC Magazine，12月号，280-281pp.

▶ X68000マシン語入門

サウンド機能研究第1回。OPM ドライバの使い方について。——高橋雄一，マイコン，12月号，181-186pp.

▶なんでもQ＆A

CARD PRO のデータを DATA PRO に転送するには？　など。——編集部，マイコン，12月号，408-409pp.

▶ Tiny FORTH

'87年9月号発表の88用を機能拡張したもの。C で記述してある。付録のディスクに収録。——小笠原博之，I/O，12月号，167-181pp.

▶ファイルセレクトコマンド

カーソルでファイルが選べるユーティリティだ。これも付録のディスクに収録。——市原昌文，I/O，12月号，208-211pp.

▶ SFORM

複数セクタにまたがるファイルを高速にシークできるようフォーマットするプログラム。——柴田直樹，I/O，12月号，235-237pp.

▶ AV WORKSHOP

第1回サイクロン CG 大会のレポートと作品紹介。新作ソフトは"DiSS-P"。——編集部，ASCII，12月号，385-392pp.

▶ JET-CACHE

HD，FD へのデータアクセスの効率を高めるディスクキャッシャ。——篠原淳一，ASCII，12月号，409-411pp.，プログラム476-485pp.

ポケコン

PC-G801

▶誌上公開質問状

PC-G801で「■」(CHR＄135) を表示させるには？　オリジナルキャラクタを作成する方法は？　G801同士をつないでプログラム，データ転送をする方法は？　PC-1600K の仕様，接続可能なプリンタ，PC-E500と PC-1480U の違いなどについて説明解説している。——編集部/ドラゴン，マイコン BASIC Magazine，12月号，63-64pp.

PC-1600K

▶ポケットコンピュータ活用研究

メモ，電話帳，時計，カレンダーときて，今回はスケジュール機能の追加だ。——塚田洋一，マイコン，12月号，311-315pp.

PC-E500

▶ポーカー

ポケコンでカードゲームを。——Es，マイコン BASIC Magazine，12月号，179p.

▶ F1 TOP LAP

5つのコースを F1で駆け抜けろ！——せとけん，I/O，12月号，250p.

シミュレーション発想

シミュレーションという言葉からは，飛行機のフライトシミュレータなどが彷彿されるだろうが，コンピュータが高性能化した現代では，ビジネス，アート，未来計画などさまざまなことが「模擬実験」の対象とされている。著者は，MIT メディアラボの客員教授を務めており，産業界における事例以外に，同研究所での数々の試みも紹介されており内容自体は豊富。ただ惜しむらくは，個々の事例をもっと詳しく解説してほしかった。

千葉正毅著　ダイヤモンド社　☎03(504)6517
四六判　253ページ　1,600円

コピーって？　ゼロックスですか

パソコンがこれだけ普及しているにもかかわらず(だからこそなのかもしれないが)，まったく使い方を知らない人は意外と多い。本書は，筆者が実際にパソコン初心者を通して得た体験をもとにユニークに書かれた初心者のためのガイドブックだ。たとえば，売り場の人にすすめられるまま買ったはいいが，何をするソフトだかわからない，といったときの対処法まで載せている。気分転換にも最適な本だ。

宇田川一彦/仲村正　技術評論社　☎03(225)3293
B6判　222ページ　980円

書籍の価格は消費税込みです

QUESTION and ANSWER

僕はX68000 ACE-HDを買ってまだ2カ月足らずの未熟者です。いままで日本語処理など，とてもとてもというような8ビット機を使っていた僕はASKによる日本語処理に感動しマニュアルを見ながらいじっていました。ところが僕の姓である「山野」という文字が学習されません。つまりいったん電源を入れ直し，もう一度ASKを起動したときに「山野」という字がいちばん初めに出てこないということです。僕が試した限りではほかの文字ではこんなこと（学習してくれないこと）は決してありません。このことを友達のEXPERTと付属のHuman68k V2.0でも同じ症状が出ました。ということは日本中に出回っているX68000はみんな「山野」という文字を学習しないのではないでしょうか。　神奈川県　山野　和也

日本全国の山賀さん，海野さん，木谷さん，呉羽さんお待たせしました。さて，試してみると「山野」以外にも「高野，宮野，前野，河野，川野……」といった名前がうまく学習されないことがわかります。これを見ても明らかなように名前が「の」などの読みで終わるとうまくいかないのです。

たとえば「川野」という文字を変換するとしましょう。この場合「かわの」の読みで「川野，河野」の2つの姓がありますが，これらは何度変換しても，一度電源を切ると第6候補より上に表示されることがありません。第1から第5候補までを見ると「川の，河の，皮の……」といった文字が表示されています。

これらは私の辞書に名詞として登録されている「川，河，皮，革，側」に“の”をつけたものです。つまり「かわ」と読む名詞が10個登録されている辞書だったら，第11候補より上に「川野」もしくは「河野」が表示されることがないということを意味しています。

ASKではこれらの名詞に“の”をつけた「かわ＋の」の読みと，「かわの」の読みを区別して学習させるらしく，それが原因のようです。これは“の”以外の格助詞についても同様です。まあ，ASKの欠陥ともいえるものでしょう。実際，「かわの」を「川野」と変換したあとには第6候補から「川野，河野」，「河野」と変換したときには第6候補から「河野，川野」と表示され，「かわの」という読みの中ではちゃんと学習されていることがわかります。

というわけですが，持ち主の名前を学習してくれないX68000なんて悲しくなってしまいますね。いろいろと試行錯誤した結果，とりあえずの解決法を見つけましたので，今回はそれを紹介しましょう。

まず最初にDICM.Xを準備します。DICM.XはシステムディスクのBINディレクトリに収まっています。付属のワープロのユーザーズマニュアルに使用方法が説明してありますので，それを参照してください。この山野さんの場合ですと，まず名詞として登録されている「やま」をすべてDICM.Xを使って削除します。その際，削除した文字をメモしておいてください。次にいま削除したばかりの文字を今度は再登録します。そのとき品詞の種類を名詞とするのではなく，単漢字にしてください。ここがポイントです。

このようにするとちゃんと山野の文字を第1候補として表示するようになります。しかし，このようにすると「やまのうえ」，「やまにはいる」などを変換すると，辞書の学習状態によっては，「山之上」，「山二入る」などと変換されてしまうので，あらかじめ「の」，「に」などの助詞を名詞として辞書に登録しておいたほうがよいでしょう。これはASKの操作性を上げるもっとも基本的な操作です。

リスト1

```
  10 int fn
  20 str cr:cr=chr$(13)+chr$(10)
7000 /*    for *.opm       */
7010 func opm_trk(i,j;str)
7020  fwrites("(t"+str$(i)+") ",fn)
7030  fwrites(j+cr,fn)
7040 endfunc
7050  func opm_vset(i)
7060  int k,l
7070  fwrites("(v"+str$(i)+",0",fn)
7080  for k=0 to 4
7090    for l=0 to 10
7100     fwrites(","+str$(v(k,l)),fn)
7110    next
7120  next
7130  fwrites(")"+cr,fn)
7140 endfunc
7150 func opm_init()
7160  fn=fopen("@@@.opm","c")
7170  fwrites("(i)"+cr,fn)
7180 endfunc
7190 func opm_alloc(i,j)
7200  fwrites("(m"+str$(i)+","+str$(j
)+")"+cr,fn)
7210 endfunc
7220 func opm_assign(i,j)
7230  fwrites("(a"+str$(i)+","+str$(j
)+")"+cr,fn)
7240 endfunc
7250 func opm_play()
7260  fwrites("(p)"+cr,fn)
7270  fcloseall()
7280 endfunc
```

毎月掲載されているミュージックプログラムを打ち込んで聴いているうちに，気にいった曲をEDなどを使っているあいだBGMにして流そうと思い，Cコンパイラを使ってX形式の実行ファイルを作りました。そのファイルをAUTOEXEC.BATファイル中で実行するようにすると，ちゃんとBGMが流れるようになるのですが，実行時に一瞬BASICモードに入ってしまい，あまり見栄えのよいものではありません。このようにならないような方法はないものでしょうか。

静岡県　森高　徹

私もOh!X LIVEに掲載されたメタルホークが気にいってしまい，メタルホークのBGMとともにHuman68kが起動するようなシステムを使っています。ま，それはいいとして，普通にコンパイルしたままだと森高さんのいうように一瞬BASICモードに入ってしまい，あまり美しいものではありません。まず，これをなくす方法をお教えしましょう。内容としては12月号の荻窪氏によるBASICでコマンドを作成する方法とまったく同じです。

たとえば，metal.basというファイルをBGMとして流したいとします。これからの説明をわかりやすくするためにAドライブにCコンパイラ，Bドライブにmetal.basが入っていることを前提として話を進めていきます。まずコマンドモードの状態（A>で入力待ちになっている状態）から，

　BC　B:metal.bas B:metal.c

とします。こうすると拡張子がmetal.cのファイルネームでBドライブにCのソースファイルが作成されますので，それをEDなどのエディタで読み込みます。そしたらb_init( ):と書かれた行を削除しb_exit(0):と書かれた行をexit(0):にしてください。エディタの検索機能を使うと便利です。

このように変更したソースファイルをディスクに保存したら，

　cc b:metal.c /W

で自動的に希望どおりの実行形式のファイルを作成してくれます。

と，これでもよいのですが，コンパイラまで持ち出すのはなんか大仰ですね。次にFM音源ドライバが直接実行できる* OPMファイルの作り方を解説しましょう。といっても，そんなに難しいことではありませ

ん。BASICのMMLを少し手直しするだけでよいのです。

まず，リスト1の関数群を音楽プログラムに加え（7000行以降），“m_”で始まる命令を“opm_”に書き換えます（エディタで置換）。要するに音楽用関数に渡されるデータをそのまま横取りするわけです。

ただし，BASICではm_vset( )のように配列全体を引数に取る関数は記述できませんので，m_vset( )の第2パラメータは大域変数をそのまま利用します。よって，opm_vset( )関数の第2パラメータは削除します。つまり，

m_vset(i,v) → opm_vset(i)

のように変更してください。

次にプログラム先頭の変数宣言部にfnとcrを加えればできあがりです。実行すると“@@@.opm”というファイルが作成されます。

演奏方法はコマンドモードから，

copy @@@.opm opm

です。これは，

copy beep.sys pcm

でpcm音が聞けるのと同じような感じですね。このサンプルでは音色定義用の配列名をv(4,10)と仮定していますので，この部分は適当に変更してください。

*.opmファイルは，コンパイルした場合に比べてサイズも小さく，エディタで簡単に修正できるなどの利点があります。

X-BASICで顧客管理プログラムを作成しているのですが，プログラムの実行途中で漢字での入力を促す場合，X68000に精通している人ならともかく，初心者ではCTRL＋XF1で漢字入力モードに入ることを知っている人はまずいないと思います。またX-BASICに精通していてもASKの環境ファイルによって，キーの割り当てが変更されていたりするとお手上げです。N(86)-DISK BASICにはKINPUTとかいう自動的に漢字入力モードに入るINPUT命令があったような記憶があるのですが，X-BASICにもこれと同じような機能を果たす命令がないものでしょうか。

神奈川県　福田　雅之

同様の質問がほかにも2，3通送られてきました。X1turboにはKEY0ステートメント中で漢字入力モードに入るようなコードがあったのですが，X-BASICのマニュアルを目を皿のようにして探してみても，それと同じような命令は見当たりません。

ですけど，Cコンパイラ付属のプログラマーズマニュアルにはそれに関係あることが説明されています。それによるとHuman68kにASK68K.SYSをデバイスドライバとして登録するとDOSコールが拡張されて（これをFPファンクションコールという），漢字変換をマシン語からも簡単に扱えるようになります。せっかくこういった便利なものが用意されているのですから，これを使わない手はありません。

今回はBASICから使えるように外部関数を作ることにしました。FPコールの1番が漢字入力モードのON／OFFを行うもので，d0にモード番号を入れてDOSコールの$FF24を呼び出すとモードに応じた処理を行ってくれます。プログラムにするとリスト2のようになります。このソースリストをEDなどのエディタで入力してアセンブル，リンクして作成された*.Xファイルを*.CNFファイルに変更してBASICが入っているディレクトリに保存してください。次にBASIC.CNFに，

FUNC＝ファイル名

の1行を追加してください。ファイル名には先ほど変更した*.CNFの“*”部分が入ります。このように変更して起動したBASICにはkinputという関数が追加されています。この関数は引数に0を与えると漢字入力モードに入り，1から255の値を与えると漢字入力モードから抜けるようになっています。たとえば次のようになります。

```
10 str a$
20 kinput(0)/*漢字入力モードに入る
30 input a$
40 kinput(1)/*漢字入力モードを出る
```

プログラム中にコメントを記しておいたので，各自好きなように変更してみてください。

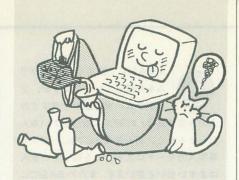
（影山　裕昭）

リスト2

```
==================  kinput.s  =======
 1: *
 2: * kinput()
 3: *
 4:     .include        doscall.mac
 5:     .include        fdef.h
 6:
 7: * infomation table
 8:
 9:     dc.l    x_init
10:     dc.l    x_run
11:     dc.l    x_end
12:     dc.l    x_sys
13:     dc.l    x_brk
14:     dc.l    x_ctrl_d
15:     dc.l    x_res1
16:     dc.l    x_res2
17:     dc.l    ptr_token
18:     dc.l    ptr_param
19:     dc.l    ptr_exec
20:     dc.l    0,0,0,0,0
21:
22: x_init:
23: x_run:
24: x_end:
25: x_sys:
26: x_brk:
27: x_ctrl_d:
28: x_res1:
29: x_res2:
30:     rts
31:
32: ptr_token:
33:     dc.b    'kinput',0
34:     dc.b    0
35: ptr_param:
36:     dc.l    fepin_par
37: fepin_par:
38:     dc.w    char_val
39:     dc.w    void_ret
40: ptr_exec:
41:     dc.l    start
42:
43:     .text
44:     .even
45:
46: start:
47:     move.l  12(sp),work
48:     move.l  work,d0
49:     tst.l   d0
50:     bne     final
51:
52:     move.l  #2,-(sp)
53:     move.l  #1,-(sp)
54:     dc.w    _KNJCTRL
55:     add.l   #8,sp
56:
57:     tst.l   d0
58:     bne     final
59:
60:     rts
61:
62: final:  clr.l   -(sp)
63:     move.l  #1,-(sp)
64:     dc.w    _KNJCTRL
65:     add.l   #8,sp
66:
67:     rts
68:
69: work:   ds.l    1
70:
71:     .end
```

質問にお答えします

日ごろ疑問に思っていること，どんなことでも結構です。どんどんお便りください。難問，奇問，編集室が総力を上げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに回答が得られるようなものも多々あります。最低限，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，直接問い合わせることもありますので，電話番号も明記してくださいね。

宛先：〒102 東京都千代田区
九段南2-3-26井関ビル
㈱日本ソフトバンク出版部
「Oh！X質問箱」係

FROM READERS TO THE EDITOR

Oh!X，そしてSTUDIO Xにも新しい年がやってまいりました。1990年はパソコンユーザーの皆さんにとってどんな年になるのでしょうか。プログラムに挑戦する人，新しいマシンを購入する人，今年も皆さんの声を聞かせてくださいね。

◆プロセッサ誌11月号の特集を読んでいた私は，10月号特集の「RISCプロセッサの設計と製作」を読んで後頭部をハンマーで殴られたかのような衝撃を受けてしまいました。ああ，価値観が根底から覆されてゆく……。

大久保　豊基（18）京都府

なんといっても，これこそOh!Xでなきゃできない特集でしょうね。でも記事を見て実際に作ったという人はほとんどいないかもしれないな。

◆“根っこ”から“初歩”そして“RISC”と順に読んでいくと「コンピュータとは」ということがとてもわかりやすく書かれていました。ちなみに私は“RISC”で理解できなくなりました。

榊　望（33）岩手県

最後の“RISC”の記事がわかれば，マイクロコンピュータに関しては無敵でしょう。

◆今月号の特集のEDSACプログラミング入門は，なかなかおもしろい。これからもこういう記事を書いてほしい。最近またプログラムリストの文字が小さくなってきた。特にLive in '89なんて，見ただけで入力する気がなくなる。

井出　博光（20）長野県

これもほかのパソコン雑誌じゃ読めない記事の代表でしょうね。編集部内でもこれは珍しいと評判でしたから。

◆世間の知ったかぶりたちによってCPUの良し悪しがビット数，クロック数，知名度によって判断され，それを見聞きした人たちがCPUがなんなのか理解しないままCPUについて語るこの頃，かえって今月の特集は新鮮に感じました。

升井　晋也（20）福岡県

◆特集楽しく読ませていただきました。やはり，1年に1回くらいはこのくらい頭の痛くなるような記事も載せてください。

石井　康介（24）大阪府

ゲーム特集の次がこれだから戸惑った方も多いでしょうけどね。これからもこういった切り口の特集はときどきやりたいな。

◆恐れるべし，斎場バンクローとジョセフソン素子。

四條　智也（19）京都府

◆斎場を出すな!!（内容はよかった）

阿部　勇次（15）神奈川県

荻窪氏の記事には毎回さまざまな反響があります。張り合いがあっていいですよ。

◆「micro Computer入門」で，最近のASAHIパソコンやEYE COMのような超入門かと思ったのですが，内容はハードウェア入門編といった感じですね（以上特集について）。特集に限らず，すみからすみまで読みがいのある内容ですが，時間がとれず全部に目を通す前に次の号が出る，の繰り返しです。プログラムの入力などもしていますので，それほど速いとはいえない私は遅れていくばかり。Cマガジンのように，ディスクを付けての発売は高くなりそうだからやっぱりいいや。でも，1,000円以内に抑えられるなら，私は賛成です。

佐藤　光（23）秋田県

最近のOh!Xは長いリストページが減って（だからハミダシが少ない）そのぶん記事が多く全部読むのは大変かもしれませんね。

◆ファンロード11月号で○H！×と伏せ字になっていた雑誌ってなぁに？

広谷　俊幸（17）広島県

まったく，それってどこが伏せ字なんですか？　でもOh!Xを伏せ字にするにはもってこいの記述方法ですね。便利そうだな～。

◆Oh!Xを買うようになって早くも1年，僕が長野県から出てきて7カ月。時のたつのは早いというけど，いったいいままで何をしていたのだろうか？　Oh!Xの中身は美味しくなっているのに僕の頭は化石になっている。誰か僕にX68000くれないかな～。

丸山　俊司（19）神奈川県

要するにX68000が欲しいわけですね。

◆宝くじの引き替え券もらってきました。よし，今年こそ1億円当ててX68000を100台（周辺機器やソフトもセットで）買って学校に寄贈してX68000ユーザーを増やしてやろう（1台は僕のものにします）。いや，それより学校の前にXシリーズ専門店をたてたほうがいいかな？

渡辺　裕之（16）北海道

なんと殊勝なこころがけ!?

◆サイバースティックに続いてサイバーノートか。きっとシャープはサイバーシリーズを作ろうとしているに違いない。これで，次のX68000の名前はわかりましたね，みなさん。それはもちろん「X68000Cyber」だ！　さあ，みんなで叫ぼう。「X68000Cyber！　X68000Cyber！　X68……」（けっこうあぶない）。

安永　吉德（20）長野県

サイバーって結局どういう意味なんでしょうか。よくわからないんですが。

◆本屋に行って，ふとOh!Xを見たら，CGAのDōGAシステムについて出ていたので，「これは人間の動きの研究などに使えるのではないか。パーソナルコンピュータもいろいろ使える道具になったなあ」と思い，X68000 EXPERT-HDを買いました。

安沢　光男（39）神奈川県

アニメ作品が主目的じゃなく，まったく別の用途に使ってもよさそうですね。

◆冬のボーナスでカラーイメージスキャナを買う予定です。やっと，やっと家に来るのかと思うと涙，涙。18万8千円飛んでっちまうよーバッキャロー もっと安くしろーなんて叫んでしまいそうです。こんどはカラーイメージジェットだ──っ。

魚住　雄一（20）愛知県

おっと，なかなか羨ましい環境を目指して

◀西村　武（18）愛知県
こっこれはスクープだあ！　しかも写真が傾いているのがリアリティを増しているぞ！　あぶない福袋を知らない人，ごめんなさい。

◀杉本　秀昭　宮城県
83年といえば，読者の半分以上がMZ-700ユーザーという全盛期でしたね。以来7年間，きっと大事に使ってきたんだろうなあ。

いますね。

◆10月15日の第2種情報処理技術者試験の自己採点で，午前85点，午後100点だったので，X68000 ACE-HDを買ってもらうことができた。しかし，もしも記入ミスなどで落ちていたら……。まあ，今はそんなことは考えずに遊べるだけ遊んでおこう。　首藤　誠二（21）大分県

とりあえずはおめでとう。えっ？　X68000を買ってもらえたことですよ。あとのことは知ーらない。

◆どっかのゲームメーカーさん，ATARIの「スターウォーズ」をX68000に出してください。

安岡　毅（15）京都府

はっきりいって私たちも欲しいです。へたにグラフィックなんかいじらないで，あのワイヤーフレームのままでいいんですから。

◆僕の友達が，あの電波新聞社が発刊しているマイコンBA○○○ Magazineに投稿し，なんと載ってしまいました。そいつは，学校で自慢話をたっぷり僕に聞かせてくれました。そこで僕がOh!Xに載ってやろうと考えたわけです。編集部の皆さん，僕に栄誉をください。けっして邪道な気持ちはありません。

荒井　信親（16）東京都

載ったからといって栄誉かどうかは知りませんよ。ただのさらしものだったり……。

◆先月，猫を飼い始めたと書いた者ですが，おかげで「猫とコンピュータ」を一番に読むようになってしまいました。うちの猫も紙にすごくこだわります。特にひどいのがティッシュペーパーで，普通の状態（あの，頭を少し出している）で置いておくと，箱から引っ張り出してそれをちぎるちぎる……。あたりが細切れのティッシュの海になってしまいます。それから，新聞を読んでいると邪魔をしにやってきて，上に寝そべってしまうといった……。これって猫の雑誌じゃなかったんですよね。失礼しました。

松野　裕之（22）徳島県

うーん，よくは知らないけど猫の雑誌だと思っている人もいるんじゃない。

◆「維新の嵐」をボクにください！　お願いします！　ぐれてやるぅ！　なんて……ね。今度Oh!Xの事典を作りませんか？　その筋の用語に解説をつけて……。神様，仏様，Oh!X様！　私に「維新の嵐」をめぐんでください。

木藤　将也（14）埼玉県

そういえば以前その筋事典というのを載せましたね。うんうん，とかいってプレゼントの話題には触れなかったりして。

◆"シャッフルパック・カフェ"完成版の絵は，私の友人伊藤君が燃える闘魂で描いたものです（試作品のほうは違うので，念のため）。ぜひ一度プレイしてください。年末は，X68000用にいっぱいゲームが出ますね。ゲームネタのイラストを描く者としては，うれPーんだけど，イマイチ欲しいソフトがないなぁ……。来年は私一押しのゲームが出る予定です。ムフフ……（カラーイラスト友人と一緒に載ってました。ドモアリガト）。　丸藤　俊之（21）神奈川県

◀小林　貴洋（17）千葉県
これってかなり以前から届いていたんだけど，X1にはイースIIIが出ないものだから……。ただ，知らなくても雰囲気はわかるでしょ。

◀寺門　修司（18）兵庫県
なるほど，黒装束は魔法使いの証ですか。でもこの絵のキャラクターはカワイイですね。どんな魔法がかけられるのかな。

えーっ，一押しのゲームって何？　あなたは何を知っているの？　えっ，おおそうかなるほど，きっと……でしょう。ムフフ。

◆'85年6月号で1回お休みして，7月号から休載になった「PUZZLE BOX」はどうしたんですか〜？　「内容一新を図り，鋭意準備中」という言葉を忘れたんじゃないでしょうね！

山下　貴幸（19）北海道

もちろん覚えていますって。なつかしいなあ。えっ，そういうこと聞いてんじゃない？ごめんなさい。

◆僕はコナミ通だから「A-JAX」のためにX68000を買うぞ!!　東野　貴士（15）奈良県

パチパチ。X68000は着々と10万台に近づいているぞ〜。

◆X68000を買って7カ月，アセンブラもマスターできない自分が悲しい。どうかアセンブラをやさしく教えてくれる記事を載せてはくれないだろうか？　ちなみに私はX68000を買う寸前まで元祖PC-6001（パピコン）を使っていたので，MS-DOSどころかVSもまともに使えなかった。初めてX68000のスペハリを見たときは，うぉ〜っと思ってしまったくらい速かった。今いちばん知りたいのはIOCSコールのラスター走査の方法で，DARIUS IIのような背景を表示してみたい。　松原　優一郎（18）千葉県

◆X68000マシン語プログラミングは最高ですね。題材も，ペースも初心者にとってもちょうどいいし。今後もいろいろなテーマを取り上げてなが〜く連載を続けてください。

鯛　富之（27）神奈川県

村田氏の連載は初心者にはちょっとヘビーな分量ですが，それだけ十分な解説がなされていますから読めば読んだだけ力になることでしょう。

◆ようやくX68000を手に入れることができた。使って感動してしまった。使いやすさに驚いてしまった。X68000を手に入れる前はN○○の8○○○mkIIを使っていたが，ほぼゲームマシンと化していた。BASICでちょっとしたプログラムを組むことぐらいしかできなかった。今考えると実に大きな損失だった。だが，X68000は違う。市販ソフトを買ってきていじるだけではもの足りない。何か自分でやらないといけないような気にさせてくれる。かといって，今の時点ではX-BASICを覚える気にはなれない。理由はひとつ，遅いからである。たとえば，3月号のマシン語入力ツールなどは，CRCチェックをONになんかしておいたときには，とてもやってられない。だから僕はいきなりアセンブラを勉強しているのである。こんな僕はバカだろうか。それはさておき，今の僕にとってX68000マシン語プログラミングの記事は最も重要な記事なのである。　山野　和也（19）神奈川県

X68000って自分も使う努力をしないとマシンに対して申し訳ないという気にさせる不思議な魔力をもっているんですよね。

◆今月のマシン語カクテルはよかった。さすがは西川さんだ。X1で3重スクロールするし，オーバー処理付きときた。これからも，がんばってください。　佐藤　史彦（17）北海道

善司ソフトの西川君はわけのわからない男だけど，本当の西川氏はごらんのとおりなかなかの実力者なんですよね。

◆受験だからOh!Xを買うのをSTOPしていたのに，悪友が「3重スクロールだって！　ホラホラ」なんて見せびらかすからおもわず買ってしまった。連鎖反応でバックナンバーも買いたくなった。ぎゃー!!　せっかく受験生してたのに！　しかし，こんなことに心を乱されてはいけない！　東京の大学に合格して日○ファ○コ○でアルバイトしてマシン語おぼえて……X1 turboZの灯を消してはならぬのだー！

石川　泰之（18）岡山県

うーん，やっぱり受験生には目の毒だったかな。でも，あと少しで春は来ます。がんばってください。

◆12月号の予告を見て私は「LIVE in '89, Beyond the Galaxyだとー！　もうかなり完成してるのにぃー」とおもわず私しかいないアパートの部屋で声を出してしまいました。でもまあ，これはしかたのないことですね。ところで，投稿したプログラムは不採用になっても永久にほかに発表することはできないのでしょうか？これはぜひ知りたいです。

安藤　正洋（20）青森県

確かに「これ以上待ってもOh!Xじゃ載りそうにないから，他誌に」と思うこともあるでしょう。そんなときは電話でご連絡ください。

◆X68000のラップトップがなかなか出ないのでついに2台目を買ってしまった。家と職場の2カ所にあると，効率がぐーんと上がりますから。我が愛妻に感謝，カンシャ，かんしゃ……。

吉田　公一（31）埼玉県

X68000が2台！　すごい，すごすぎるーっ。

◆X68000さんへ。ノーライフキング出演おめでとう！　次回は主演をめざして頑張ってください。　近藤　周司（19）神奈川県

んだ，んだ。

◆オブ・ラ・ディ，オブ・ラ・ダがビートルズの曲だとは……。またひとつ頭がよくなってしまった。買ったその日に打ち込んで「ああ感動だ!!」とか思ってしまった。この曲はたしか運動会か何かで流れていたように思ったけどな〜!!　黒崎　雅己（18）千葉県

ビートルズのヒット曲といってもほとんど黒崎君が生まれる前の話だものね。うーん，世代の違いを感じてしまう。

◆どーでもいいことだが，いつのまにか飯島真理サンが結婚しているぢゃあーりませんか。しかも外人サンと。うーむ，デカルチャー。

龍尾　謙二（23）岐阜県

「きゅーん，きゅーん……」なんて歌っていた飯島真理さんが……，そうなのかぁ。

◆DōGAを買ったので，買い始めた（たぶんこれから毎月買うと思う）。X68000 PROをやっと買ったのだ。もう88SRはE-500のプログラム保存用としてしか使わなくなるだろう。学校が忙しく，X68000をいぢるひまがない。本読むひまもない。2M増設RAM欲しいけど売ってなかった。どーでもいいけど“ジェノサイド”を連射装置を使わずにコンティニュー9回（のみ！）でとうとう終わらせた（約90分で）。実はうれしい。

三森　浩一（20）東京都

◆いまC言語でプログラムを作っています。しかし1〜2時間程度で目が痛くなって，ジェノサイドをエンディングまでやるのはヘンでしょうか。なぜかゲームだと体の調子もよくなってきます。仲間うちではゲーマー杉島と呼ばれますが，その呼んでいるやつが1日ゲームをやらないと手がふるえるという人物です。いったい何と呼べばいいのでしょうか。

杉島　俊樹（20）愛知県

禁断症状ですか……，でもそういうあぶないお友達がいたほうがパソコンやっていっても面白いでしょう。

◆ふとわれにかえるとゲームをしている自分。こんなことではいけない，××も○×も△□もしなければ……。しかしついついゲームディスクに手がいってしまう。そこで名づけて「初心者のための正しいX68000入門」というのを大々的にやってほしいです（私はX68000を9月28日に買いました）。累計出荷10万台記念として1月号ぐらいでおねがいします。

大森　康雄（22）滋賀県

今月のOS特集を参考にしてくれるとうれしいな。ゲームユーザーにもおいしいオペレーションがあると思うんだけど。

◆初めてマシン語で画面に文字を出すことができました。うれしー。これからもっと勉強したいと思います。いやー，それにしてもMZ-80KのマシンランゲージSP-2001の解説書が今頃役に立つとは。ソフトテープはとっくにウニとなっているのに。　西村　武（18）愛知県

おおーっ，SP-2001！　スゴイ！

◆Oh!Xでサブルーチン単位のプログラムコンテストをやってもらえませんか。毎月，課題と仕様を決めてプログラムを募集し，速さやバイト数を競うのです。できれば，その応募作品をPDSにして読者が自由に各々のプログラムに利用できるようになったら素晴らしいと思うのですが。

松尾　哲（21）京都府

◆MZシリーズ（700/2500以外）の投稿はないのでしょうか。ショートでもよいのでぜひ載せてほしいものです。　岩谷　政治（18）愛知県

S-OSなら結構来てるようですよ。

◆X68000を買ってもう5カ月たちましたが，持っているゲームの半分ぐらいはシューティングゲーム。それというのも，それ以外のジャンルに超本格的といえるゲームが少ないからだと思います。確かにX68000はシューティングゲームに向いていると思いますが……。

田中　義人（20）福井県

そうですね。でもアドベンチャーとかもいいのがありますよ。そうそう，付録のカタログも見てくださいね。

◆「猫とコンピュータ」はホンニャアの行動がしっかりと，描かれていてとても面白いのでいつも読んでいます。ところで，X1Fの6MHz化に成功された方がいらっしゃるそうですが，是非ぜひOh!X誌面上に載せてください，お願いします（う〜ん，これでMZ-2500に馬鹿にされなくなるかもしれない。たとえOh!X誌上を埋めつくそうともX1を大改造しなければならなくとも，真のユーザーなら涙を流して喜ぶ!?）

永井　雅晴（17）愛知県

◆ハッハッハッ，私のX1turboZはもうすぐ8MHzだ!!　今は4/6MHzの切り替えだが，8MHzにするとソフトの問題があるんですよね。特にディスク。あとDMAを使ったディスクアクセスをするソフトがいまだに動かないんですよ……。困ったもんだ……。

福井　利夫（17）香川県

◆大阪の宮本さんのX1Fは6MHzだとか（11月号参照）。私のX1turboIIは8MHzノーウェイトです（さすがにCPUはZ80Bにしてあります）。残念ながら8MHz動作時にはFDDとのシンクロがシビアになり，データ化けを起こすことがあるのでもっぱらturbo RAY TRCERの計算時間短縮くらいにしか使えませんが……。6MHzモードもあるのですが，最近はMIDIを楽しんでますんでノーマルモード（4MHz）です。うーん。

山口　幸一（23）東京都

クロックを上げている人って結構いるんですね。もちろんそれなりに苦労はされているようですが。

◆パソコン歴4年と10カ月（X1turbo）ですが，BASICの勉強をしただけで，パソコンテレビのテレビとしてしか使ってませんでした。しかし，今年はちと違う。せっかく1985年1月号のOh!MZから全部買ってたまってるんだから，S-OSやマシン語体操1・2・3，試験に出るX1，etc.で勉強して，最近X68000の影にかくれがちなX1を盛り上げたいと思います。

窪　祥宏（21）富山県

よーし，みんな，いまこそX1ユーザーの力を見せるんだ〜。

◆CPUのビット数について，最近新しい規準を考えた。それは，インストラクションコードの最小単位をもってそのCPUのビット数とするというものである。この規準だとZ80，8080，6809等は当然8ビット，MC68000系はすべて16ビット，そして8086系は80386を含めて，なんと8ビットになるのである！　大笑いである。だから.速いという説もある（コードが短いほうが速い）。1byte＝8bitsは必ずしも成立しないと聞いて耳をうたがい，上式を世にはびこらせたのはモトローラだと聞いて，うなってしまった。

中内　英裕（25）東京都

うーん大胆な分類だなぁ。

▲岡村　直也　兵庫県
数少ないX1用新作ソフトのなかにも，結構いいものがありますから。岡村君はこのローグアライアンスにしっかりハマってるんですね。

『ヴァリスCLUB』って知ってる？
前巻に続いてまた買ってしまった。
冗談の様なビデオだけど発行は日本ソフトバンク。
うーん…
Coming Out!! ヴァリスII

▲味野　真一　岡山県
やっぱり来ましたねぇヴァリスネタ。かわいく描けているので○。でも，ビデオの「ヴァリスCLUB」についてはOh!Xは知らん顔しよっと。

◆本体はX1Ck，X1turbo40，X1turboZⅢと変わりましたが，ディスプレイはCZ-800DSのままです。そうです！　今は“ビデオ入力”で画面を見ているのです。それにしてもX1turbo40，X1 turboZⅢとも後ろのインタフェイスの数が少ない。X1turboZⅢはデジタルRGBもCMTインタフェイスもない！　今度新機種が出たら，今度はプリンタインタフェイスか？　それとも……？
井原　崇（24）宮城県

X1turboZⅢでは結構いろいろとなくなってますね。

◆最近，昔のテープ版ゲームをディスク版に改造するなんてことを始めました。ディスクアクセスルーチンじゃX1turboZⅡのバンクメモリを使ったおかげでメインメモリをあまり食いません。最終目的はザナドゥをディスクに詰め込む事だぁ，と息まいていますが，無理かなぁ。これって結構面白いですよ。今をときめくクリエイティブ・マシンX68000といえどもこんな事はできないでしょ。だって，テープ版ソフトがないんだもん。　伊藤　雅彦（20）東京都

テープ版のゲームだっていいものはたくさんありますからね。ボコスカウォーズなんていまでもときどき遊びたいと思うもの。なつかしいな。

◆ここで一句。富士通よ　俺にTOWNS　一円で!!（時事季語：1円）
大村　邦嘉（18）神奈川県

もう，こまりますねぇ。

◆PC-8801MCには絶句した。しかし，20万円以下でパソコンを出すなら8ビット機はまだ生きのびてもらわなくてはメーカーも困るだろうからこの動きは当然なのかもしれない。もしも，MCがヒットしたならば「8ビット機は周辺機器でよみがえる」という原則が確立する。CD-ROM内蔵のX1twinや光磁気ディスク内蔵のX1turboZが出てくるかもしれない。でも僕はそんな未来は見たくない。　田中　義教（29）東京都

とりあえず，X68000クラスのマシンが20万円ぐらいになればそんな心配はないんでしょうけどね。

▲三浦　崇　長野県
うちの雑誌じゃパソコンが買えるかどうかがよく話題になるけど、車をぽんと買っちゃう人だっているんだなあ。考えてみると普通のことだけど。

ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目（売る・買う・氏名・年齢・連絡方法……）を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集室では責任を負いかねます。
●応募者多数の場合，掲載できない場合もあります。

仲間

★オリジナルソフト制作を中心に活動している，「FELMIA soft」では幅広くスタッフを募集中。内容は，ファンタジー系を主とするゲームソフトなどを制作し，コミケなどで販売するほか，パソコン関係の会誌を発行したり，音楽を作ったりしています。開発室は滋賀県大津市にあり，会員の方は豊富な機材と資料を自由に利用できます。興味のある方は62円切手を同封してご連絡ください。詳しい案内書をお送りします。マイナー機種歓迎。入会金不要。〒520 滋賀県大津市二本松3-5　中島能和（19）

★X68000ユーザーを対象としたサークル「INSIDEWOOD68K」では会員を募集します。主な活動内容は月1回の会報発行です。詳しくは62円切手を同封のうえ連絡ください。〒515-04 三重県多気郡明和町佐田451-14　中井宏行（17）

★X1turboユーザーを対象とした「Turbo Bulletin」では会員を募集しています。活動内容はディスク会報発行，PDSの配布などです。入会ご希望の方は62円切手同封のうえ封書にて連絡を。また120円切手2枚で会報をお送りします。〒729-01 広島県福山市今津町3-33-103　竹田智哉（18）

★X68000ユーザー対象のサークル「LUM'S COMPANY」では現在会員を募集しています。月1回のディスク会報での交流をしています。合同作成にあたり，アセンブラに詳しい方，グラフィック（ゲーム，デザイン）のできる方は特に歓迎します。「やってやるぜ！」という人は62円切手を同封のうえご連絡ください。〒711 岡山県倉敷市児島小河9-7-3　伊藤智則（17）

★「S-OSユーザーズ・クラブ」では新規会員を募集します。当会の活動内容は，1.ディスク会誌の発行，2.オリジナルプログラムの交換，3.パソコンネットワークやミーティングによるS-OSに関する情報交換などです。入会を希望される方は，住所氏名使用機種を下記までご連絡くださるか，「S-OSネット大阪」☎06(969)8459までアクセスください。折り返しディスク会誌「S-OS通信」をお送りします。〒536 大阪府大阪市城東区鴫野東1-13-18　森喜一郎（24）

売ります

★プリンタCZ-8PC1＋リボン＋用紙を2万円程度で。松下・ワープロFW-K101（9インチCRT，3.5FDD）＋リボン＋用紙を4万5千円程度で。プリンタ，ワープロともにマニュアル，箱，付属品あり。連絡は往復ハガキで。〒651-11 兵庫県神戸市北区鈴蘭台南町4-5-6　七浦啓有（18）

★ローランドCM-32Pを4万5千円で売ります。付属品はすべてあり。連絡は往復ハガキで。〒935 富山県氷見市柳田2286-2　岩上章（18）

買います

★X1turbo用増設ドライブCZ-51Fを送料込みで1万2千円で（箱，マニュアルのどちらかがない場合は1万1千円，両方ともない場合は1万円）。完動であれば傷，汚れは可。連絡は往復ハガキで。〒236 神奈川県横浜市金沢区高船台2-47-5　高橋誠司（20）

★X1用FM音源ボードCZ-8BS1を1万円以内で。多少のキズ可。完動，付属品，説明書付きのもの。連絡は往復ハガキで。〒364 埼玉県北本市西高尾5-12　滝瀬至永（19）

★YAMAHAのリズムマシンRX-7か，ローランドのリズムマシンR-5を3万円前後で（マニュアル，付属品付きで，傷，汚れは可）。それとYAMAHAのミキサーMV100を1万円前後で大至急お願いします。連絡は往復ハガキで。X1でMIDIをやりたいんです。〒299-25 千葉県安房郡丸山町加茂860-1　山田正浩（16）

★X68000用MIDIボードCZ-6BM1を1万5千円。ローランドの音源MT-32，D-110，U-110のいずれかをそれぞれ2万8千円，4万円，4万8千円で。送料はこちらでもちます。各マニュアル付きで。連絡はハガキで。〒380 長野県長野市東之門町2549-3　大石壮吾（17）

★KAWAI K1Ⅱを6万円，同じくKAWAI PHmを2万円で。完動，付属品，説明書付きならば多少の傷や箱なし可。往復ハガキにて。〒183 東京都府中市多磨町1-34-7（株）ジャコム多磨寮　山口幸一（23）

★MZ-2500用RAMディスクMZ-1R37または同等品を1万5千円前後，パレットボードMZ-1M10を7千円で，ディスプレイテレビMZ-1D24を4万円前後で。完動品なら箱，説明書の有無は問いません。連絡は希望価格明記のうえハガキで。〒280 千葉県千葉市大森町319-13　黒沢由美江（17）

バックナンバー

★Oh!Xの1988年11月号を送料込み千円で。切り抜き不可。連絡は往復ハガキでよろしくお願いします。〒675 兵庫県加古川市東神吉町神吉823-167　栗原美由紀（18）

編集室から from E・D・I・T・O・R

DRIVE ON

このコーナーでは，本誌年間モニタの方々のご意見を紹介しています。今回は，11月号の記事に関するレポートです。

●「micro Computer入門」という今回のテーマは，特集全体にはっきりと反映されていたと思います。特集の最初の「コンピュータの根っこ」では基本となる0と1を押さえており，「RISCプロセッサの設計と製作」からは，自分でものを作り出すという開発精神の原点を感じました。ソフトウェアを作るのにハードウェアの知識が不必要だという，最近の風潮はある意味では正しいのかもしれません。しかし，ソフトウェア開発においてイザッ！ というときはハードウェアの知識があったほうがいいのではないかと思います。

藤田康一（19）X68000PRO 静岡県

●入門編ともいえる「コンピュータの根っこ」は大変面白く，またわかりやすく書けていると感心しました。あれなら，中学生でも2進演算が理解できると思います。「初歩からのＣPU物語」は，ずいぶん苦労して書いているような気もしますが，よくまとまっていると思います。また，注釈が豊富で理解の助けになります。「ヘンなコンピュータ」もよかった。具体的には，ちまたでは概念だけが先走りしすぎている感もあるこれらのコンピュータをよく限られたページ数でうまく説明できたと思います。「RISCプロセッサの設計と製作」はかなり難しいけれども，コンピュータにとっての心臓部であるCPUの仕組みを身をもって理解できるということが素晴らしい。

「X68000ハードウェア操縦法」はCRTコントローラなどマニュアルが存在しない部分も詳しく解説されており大変参考になりました。

森川一（24）X1turboII，X68000ACE-HD 北海道

●「RISCプロセッサの設計と製作」は史上最低のCPUということでしたが，MC68000や80386などのCPUについて詳しくない私でも理解できる範囲の記事だったので勉強になりました。今回の特集はテーマがテーマだけに難しいですが，「EDSACプログラミング入門」から「ヘンなコンピュータ」にいたるまでひと通り説明されていたので「入門」としてはよかったと思います。ただ，「コンピュータの根っこ」は抽象的すぎてかえって2進数が理解しにくかったのではないかという気もしました。

田中実（19）大阪府

●ともすると概念的になりがちな特集のなかで「周辺LSIを使いこなそう」は具体的でよかったと思います。特に，IOCS関係で詳しい説明とサンプルプログラムが付いていたのはうれしいですね。欲をいえばもっと応用的なこともやってほしかったのですが紙面の都合上難しいのかもしれませんね。

西田宗千佳（18）X1turboII，X68000ACE 福井県

●CGAシステム自体はシェアウェアとして，多くの人から賛同されているので「DōGA・CGアニメーション講座」の連載には意義があると思います。しかし，問題となるのはどういう形式で連載するかです。CGAシステムはBASICやCなどの一般的に共通なアプリケーションとは異なり，単なる1アプリケーションです。たとえば，CGAシステムを多くの人に知ってもらいたいのであれば，マニュアルの簡単な解説から始めるとか，ユーザーとのコミュニケーションの場にするとか，質問コーナーを大きくするとかの方法を打ち出すべきだと思います。ただ，今回の内容についてはソフトを持ってない人でも，読んで理解できる部分がありました。

湯澤聡（26）MZ-2531，X1turboIII，MZ-2861，X68000，MSX，PC-6601，PC-1360K 東京都

●X68000の場合，雑誌への掲載プログラム言語は特定すべきではないと思います。掲載するプログラムはやはり用途に応じて適した言語があれば，それを使うという方針でいいと思います。今回のプログラム「ばばぬき」についていえば特にBASICにする必要はなかったのではないかと考えます。移植性なども考えてCでスマートに書いたほうがよかったような気がします。

中野賢一（29）MZ-2000，X1/G/turboII，FM-8，FP-200，PC-8801/9801，PC-1251，B16LX 山口県

●「マシン語カクテル in Z80's Bar」はなかなかいい味出していると思います。が，くれぐれも内輪話のみに走らないようにしてください。もっとも，その内輪話が面白いのですが。また，会話形式の連載講座自体は，自分たちが話しているようですし，ソフト開発のちょっとしたノウハウを学ぶにはとてもわかりやすいと思います。サンプル的に出てくるプログラムも，親しむためのものから実戦的なものになってきたし，徐々によくなってきたのではないでしょうか。今月に関していえば，ページは増えたけど，中身は充実していると思います。やはり，楽しくやっていかなければ覚えるものも覚えないと思います。こういう記事はOh!Xのような硬派な雑誌では異端児的に見られているかもしれませんが，私にとっては導きの光になっています。

大津和之（19）X1turboZ 福岡県

●今回で最終回の「MZ-2500グラフィックエディタ作成講座」は，MZ-2500用のソフトの発売本数が非常に少なくなった現在では，とても意義があったと思います。また，MZでここまでやれるというのがわかっただけでも非常に意義があったのではないでしょうか。

高田博（31）X68000ACE-HD，PC-8001 和歌山県

●「THE SOFTOUCH」を書く人それぞれで，一概にはいえないけど，どうも紹介するソフトの選び方に偏りがあるように思います。ときには，ほとんど売れないようなものや高いものなども紹介しても面白いでしょう。紹介の形式自体はいいと思います。

大山栄一（16）X68000ACE-HD 大阪府

●今月の「LIVE in '89」のMETAL HAWKはよかった。前代未聞の25音音色定義，音色設定，トラックへの書き込み，ポルタメント，すべて関数。おまけに編集部の付け加え。すごくいい，最高だった。ただ，曲自体とは関係ないが，初心者が打ち込む場合の説明がほしかった。たとえば，2620行まで打ち込んだ時点で2600行と3780行を打ち込んで，270行をMUSIC( )にすれば音楽が聞けるとか。そうすれば，1トラックずつ聞くことができたのに。

原田謙（15）X68000PRO 広島県

ごめんなさいのコーナー

11月号 METAL HAWK

P. 142 12月号での訂正に誤りがありました。12月号のこのコーナーで修正した"> b"の部分を"< b"に変更してください。

12月号 X68000マシン語プログラミング

P. 97 サンプルリスト4-bのスタックフレームのためのレジスタ復帰部（13行）が誤っています。適当に直してください。

その他，これまで連載中にいくつか誤りがありました。詳しくは本文のほうをお読みください。

バグに関するお問い合わせは
☎03(230)7683(直通)
月～金曜日16：00～18：00

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます。入力法，操作法などはマニュアルをよくお読みください。

また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません。ご了承ください。

新年あけましておめでとうございます

▼クリスマス前に新年のあいさつもそらぞらしい気がしますが，これも出版界のならわしですのでご理解を。さて，今月は豪華特集2本立てです。初心者にはわかりづらいOSとパソコンの操作法を，上級者にはC言語を使った応用例と話題のGCCについて触れました。どちらが参考になりましたでしょうか。

▼年末年始といえばクリスマスプレゼント，お年玉，ボーナス，年末調整と臨時収入が多くなる時期。ここらでX68000を買ってみたけど……とか，もっとソフトが欲しいという方に朗報。これまで発表されたX68000用ゲームをまとめて別冊付録にしてみました。懐かしいゲームから最新ゲームまで，ほとんどが網羅されています。冬休みはこたつでゲームという人は活用してみてください。

▼先月の特別モニタプレゼントの影響か，連日アンケートハガキが山のように届いています。さて，プレゼントが当たるための方法などはありませんが，当たらなくなくするための方法はありますので紹介しましょう。

アンケートにはすべて回答する

応募要項にある「すべてご記入のうえ」という一文を見落としている方がたくさんいます。また，ハガキの表と裏に年齢欄があるのも手違いではありません。

派手にしすぎない

目立つハガキのほうが当選率が高いと思い込んでいる人がいるようですが，大きな間違いです。ハガキの表側まで着色したり，極度に派手に着色されたものは郵便屋さんの迷惑になりますので自動的に選外となります。

数多く出しても無駄（かもしれない）

なかには古いハガキをまとめて出してくる方がいますが，整理の都合上，見つかり次第ホッチキスで1部にまとめられます。

抽選で選ぶとはどこにも書いていない

ほぼ無作為抽出ですが，ものによってはメッセージ内容が考慮される場合があります。

これらを守れば，少しは可能性が出てくるわけですね。では幸運をお祈りします。

投稿応募要領

- 原稿には，住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。
- プログラムを投稿される方は，詳しい内容の説明，利用法，できればフローチャート，変数表，メモリマップ（マシン語の場合）に，参考文献を明記し，プログラムをセーブしたテープ（ディスケット）を添えてお送りください。また，掲載にあたっては，編集上の都合により加筆修正させていただくことがありますのでご了承ください。
- ハードの製作などを投稿される方は，詳しい内容の説明のほかに回路図，部品表，できれば実体配線図も添えてください。編集室で検討の上，製作したハードが必要な場合はご連絡いたします。
- 投稿者のモラルとして，他誌との二重投稿，他機種用プログラムを単に移植したものは固くお断りいたします。

あて先

〒102 東京都千代田区九段南2-3-26井関ビル
日本ソフトバンク出版部
Oh！X「㊢㊐㋮㊔」係

SHIFT・BREAK

▶人間いつがいちばん気が緩んでるって，トイレで用を足し終わったとき以上はない。それがドアを開けた途端にゴキブリがおっこちてきたときは死ぬほどビックリした。思わず足がそのへんをさすらってしまったほどだ。そいつは圧殺の刑に処されたが，次の日ゴミ箱からガサガサ音がしてきたときは……（あー，ゴキブリ嫌い。H.U.）

▶最近私のまわりは，ノートサイズのNECの新製品，98Nを買う，買わないの話題でもちきりである。据え置き型とまったく同じ機能を備えていれば，布団の中（冬季のみ）やトイレでなど，家のどこででも使えるという点から評価できるのだが，あくまでセカンドマシンのようである。持ち歩くだけなら，電子手帳のほうが使えるような気がする。（亀）

▶私自身にとってやさしい記事・難しい記事どちらを書くことが良いことであるのかはわかりません。でも私は，それがパソコンを買ったばかりの人であっても，パソコンのソフトでもハードでもなんでもござれの人でも，すべての読者にわかってもらえてすべての人に感動してもらえる原稿が書きたい。ちょっと早いけど（て）の1990年の目標でした。（て）

▶つい最近までもうアイドルは卒業したなと思っていたのに，友達の大学祭のコンサートで某アイドルを見にいってからハマってしまった。レンタル屋に行ってビデオは借りてくるわ，CDは借りるわで私の財布はまた軽くなってしまった。ところで最近クラシックのCDが安くなっていますから，皆さんも聴いてみたらいかがですか。（H.K.）

▶このところ，隣のアパートの取り壊し作業があって，その振動がこちらにまで伝わってきていた。それがやっと終わったと思ったら，今度はそこに新しい建物が建つという（当然か)。おかげで当分安心できない。万一，来月号のプログラムが壊れでもしたら大変だ。で，神経質かもしれないが，HDのバックアップをマメに取っている今日この頃。（A.T.）

▶編集部へ向かう途中，何気なく見上げた月のすぐ隣で見慣れない星が輝いてた。金星食の始まる10分ほど前。何日もかけてだんだん近づいていたんだろうに，いまのいままでちっとも気づかなかった自分がちょっと悲しい。空を眺めるこれっぽっちのゆとりもないような生活をしているわけじゃあるまいに。駄目だね。ああ，駄目だ，駄目だ。（Mu）

▶というわけで、ペレストロイカである。天安門である。ベルリンの壁である。あまり資本主義の亡霊に惑わされて日本みたいになってしまわないよう気をつけてほしい。創造より破壊を得意とする先進国みたいなのがこれ以上増えたらと思うとぞっとする。モノにとらわれすぎた文化は必ず滅んでいくのである。では，よい寝正月を（K）

▶先月この欄で書いたTRONチップ用のGCCからX68000用のVer1.36を作ったら，最近出回っているVer1.36よりも速いコードを出した。Ver1.36ではあの'NASI'という文字定数を許すので，ついでに2進定数を許すように改造して，BCで変換したBASICプログラムをコンパイルできるようにしてしまった。（右が鈴木，左が会田のKO）

▶そろそろたい焼きのおいしい季節だ。昔，私はその作られるさまが面白くて毎日買いもしないのにたい焼き屋に通っていた時期があった。そんなある日，お店の兄ちゃんが「これあげるから帰んな」と，たい焼きをひとつくれた。そのときは喜んだが，今ふと考えると「たい焼きひとつ買えないびんぼうなガキ」と思われてたのかなぁ，私って。（E.O.）

▶先日，大丈夫日記を見たら女の子が可愛くて思わず世界は平和などと歌いながら中華鍋でうどんを炒めた。食べながら久しぶりThe Band of Holy Joyのレコードを聞いたら泣けたので大衆酒場に行きMomusちゃんで左に跳んで右にステップした。帰ったらザ浦安から電子メイルが届いており封を切ったら怒濤が押し寄せた。気がつくとキネマな夜だった。（S）

▶「うに」「ふまん」を変換すると「UNIX」「Human」と出る今日この頃。GCCはオブジェクトのオプティマイザをつければ2000dhrystoneくらいは出そうだし，「ちょっとした前処理」を加えれば3000近いのが出るんじゃないか，というのが最近のKO氏との会話でした。現状ではPC-9801RA＋MSCの半分の性能しか出ない。ヒマな方いませんか？（U）

▶そうかX68000のゲームソフトっていつの間にやら150本近くも出ていたのかぁ，と言いつつ出来上がったのが本誌初の別冊付録です。オリジナルが少ないとかアクションばかりだとか言う人もいたけど，ずらっと眺めてみると結構いいラインナップでしょう。できれば来年以降も続けていきたいのですが……。はっきりいって作るのしんどかった。（T）

microOdyssey

私のパソコンに関しての歴史はまだ浅い。発売されたばかりのPC-9801VXをすでに10年も昔からあるかのごとく使い始めたのが最初だ。そのときから私のコンピュータについての考えは，ビジネス中心の世間の流れとほぼ同じである。つまりパソコンとはデータベースであり，C言語はアセンブラよりいいと思うなどである。

先日某氏から「パソコン少年はアセンブラと聞いただけでピクピクと心がときめく。それは本能なんだ」と聞いた。私はアセンブラと聞いても胸はときめかない。が，その気持ちはわかるような気もする。それは私が映画少年だったからかもしれない。映画少年とは，見るだけでは飽き足らず実際に映画を作る少年のことだった。まず，作ってみるという気持ちが先に立ち，撮影という言葉には心がときめいた。

社会は，価値観の異なるテリトリーのようなもので仕切られているといってもよいだろう。つまり，パソコン少年と映画少年，プログラマとカメラマンの価値観は異なる。ところが最近，各業種間の境界線がなくなりつつあるという話をよく聞く。また，これからのビジネスマンは自分と異なる世界を知れという教えもあるようだ。これらは，テリトリーを仕切っている壁が壊れつつあるということだろうか？

MITメディアラボでは，放送，コンピュータ，出版の「3つの輪」を合わせたような新しいメディアの構築を研究している。CD-I，デジタルビデオ，パーソナル新聞・TV，ブロードキャッチ，喋る机，トーキングヘッズ……。「すべてのコミュニケーションテクノロジーは合体して変貌する苦しみを味わっている。それは全体を一つのものとして扱って初めて正しく理解できる」とはMIT教授ネグロポンテの言。メディアは，喋る時代，書く時代，コピーを作り出せる時代を経てデジタルの時代になる。このようなメディアでは，映像も，音声も，匂いすら共通のデジタルデータとして扱えるようになるだろう。

映画は，音楽や写真などさまざまなことを引き込んだ総合技術である。デジタルデータを媒体としたメディアも，異なった価値観をもつ情報を同じ土俵に引き込んだ総合技術である。このようなメディアは，電話であり，TVであり，雑誌であり，パソコンであり，すべてを凌駕するものになり得る。そうして，それを作るのはパソコン少年や映画少年なのだ。それは，技術と芸術の結合という歴史でよくあるエポックメイキングのひとつといっても過言ではない。

世間では，このような複合的なメディアをマルチメディアと呼んでいるようだ。だが私が考える「マルチメディア」とは，むしろ動画TV電話に近いのかもしれない。それは「彼は怒っている」という情報ではなく「怒りの感情」そのものが伝えることができるメディアである。たとえば，「こっちは雨なのでゲームでもしよう」と電話したとき，ディスプレイに映るのは，お互いの顔，窓から見える雨模様，パーソナルTVからのニュースや映画であり，BGMは実際の雨音とCDからの音楽だ。2人はネットワーク上で同じビデオを見ながら会話とゲームを楽しんでいる。電話が開発された当初は，長電話自体を楽しむ人種など想像されていなかったと思う。きっと未来の恋人たちは，自宅，職場，旅行先からデータ交換を楽しむにちがいない。　(S)

1990年2月号1月18日(木)発売

特集　画像圧縮へのアプローチ
- アダマール変換による自然画像圧縮
- 噂のPIC.Rアルゴリズム解説

X68000用ゲーム　GONGON

MZ-700用紙芝居　Eyelarth

全機種共通システム　小型コンパイラ言語 TTC++

Oh!X LIVE in '90

MZ-2500用&X68000用オーダイン

バックナンバー常備店

東京	神保町	三省堂神田本店5F	03(233)3312
	〃	書泉ブックマートBI	03(294)0011
	〃	書泉グランデ5F	03(295)0011
	秋葉原	T-ZONE 7Fブックゾーン	03(257)2660
	八重洲	八重洲ブックセンター3F	03(281)1811
	新宿	紀伊国屋書店本店	03(354)0131
	高田馬場	未来堂書店	03(200)9185
	渋谷	大盛堂書店	03(463)0511
	池袋	リブロ池袋店	03(981)0111
	〃	西武百貨店9F コンピュータ・フォーラム	03(981)0111
神奈川	横浜	有隣堂横浜駅西口店	045(311)6265
	〃	有隣堂ルミネ店	045(453)0811
	藤沢	有隣堂藤沢店	0466(26)1411
神奈川	厚木	有隣堂厚木店	0462(23)4111
	平塚	文教堂四の宮店	0463(54)2880
千葉	柏	新星堂カルチェ5	0471(64)8551
	船橋	リブロ船橋店	0474(25)0111
	〃	芳林堂書店津田沼店	0474(78)3737
	千葉	多田屋千葉セントラルプラザ店	0472(24)1333
埼玉	川越	黒田書店	0492(25)3138
	川口	岩渕書店	0482(52)2190
茨城	水戸	川又書店駅前店	0292(31)0102
大阪	北区	旭屋書店本店	06(313)1191
	都島区	駸々堂京橋店	06(353)2413
京都	中京区	オーム社書店	075(221)0280
愛知	名古屋	三省堂名古屋店	052(562)0077
	〃	パソコンΣ上前津店	052(251)8334
	刈谷	三洋堂書店刈谷店	0566(24)1134
長野	飯田	平安堂飯田店	0265(24)4545
北海道	室蘭	室蘭工業大学生協	0143(44)6060

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は，とじ込みの振替用紙の「申込書」欄に何年何月号からをご記入のうえ，**年間購読料6,720円**(税込)を添えてお申し込みください。その際，裏面の通信欄に「○年○月号よりOh!X定期購読希望」と忘れずに明記してください。なお，すでに定期購読をご利用いただいている方には，購読期限終了と同時にご通知申し上げますので，同封の払込用紙をご利用ください。

海外送付ご希望の方へ

本誌の海外発送代理店，日本IPS(株)にお申し込みください。なお，購読料金は郵送方法，地域によって異なりますので，下記宛必ずお問い合わせください。

日本IPS株式会社
〒101 東京都千代田区飯田橋3-11-6
☎03(238)0700

Oh!X　1月号
■1990年1月1日発行　定価560円(本体544円)
■発行人　孫　正義
■編集人　橋本五郎
■発売元　(株)日本ソフトバンク
■出版事業部　〒102 東京都千代田区九段南2-3-26　井関ビル
Oh!X編集部　☎03(230)7681
出版営業部　☎03(230)7670 FAX 03(262)8397
広告営業部　☎03(230)7672
■印　刷　凸版印刷株式会社

近日発売！日コン連SOFT紹介

史上初、3Dで秒間描写30コマ以上を実現！
360度全方向への進行可能、究極の3Dドライビングゲーム

F.T.SCAN（エフ・ティ・スキャン）

対応機種：X1/X1ターボ（5インチ2D），X68000（5インチ2HD）
開発者　：Final Tear Z / Seafy・NAZ・Spark
¥5,980

《内容・特長》
・バトル、レース、鬼ごっこ各モードを備えた、F．T．SCANシステム搭載の３Ｄアクションゲーム。
・2人同時PLAY可能。通信による4人同時PLAYも可能。
・X1、X68000共サイバースティック対応。

究極美表現エキサイトX指定第２弾
本格的ファンタジーアドベンチャーゲーム

AQUARIUS（アクエリアス）

対応機種：X68000（5インチ2HD）
¥5,980
原作・開発者：神戸大学情報統計部　赤坂賢洋
グラフィック：神戸大学情報統計部　細見格・中野博之

《内容》
魔法により、石に変えられ洞窟に閉じ込められた恋人を助け出すため、オルフェウスという若者が旅する …… というストーリー。
グラフィックに究極の美しさを追求したファンタジーアドベンチャーゲーム。

冬休みの英語の宿題は、このソフトでバッチシOK！
品切れ店続出！大人気！究極のずるかましソフト！

翻訳ヘルパーずるかまし

対応機種：X68000（5インチ2HD）　2枚組
開発者/プログラム：大阪市立大学マイコン研究会　山本賢一
辞書　　　　　　　：大阪市立大学マイコン研究会　山本博之
¥5,980

《内容および特長》
▶**翻訳ガイド**・・・入力または、ＰＤＳ等で拾った英文ファイルに対して、辞書登録されている単語（語尾変化型含む）が含まれていた場合、その単語及び意味を表示またはプリントアウトします。
▶**ＴＶ英和辞書**・・・英単語または、その単語の頭文字からの任意の文字を入力すると、該当する単語の意味を表示またはプリントアウトします。また、同時にそのスペルに近い単語も６つ表示し、それらの中からの選択も行えます。そのため、スペルがあやふやな覚えの単語でも確実に意味を得ることができます。
▶**ＴＶ和英辞書**・・・入力された単語を訳として持つ英単語を順次、表示またはプリントアウトします。この時、英単語の頭文字の指定なども行え、目的の英単語をすばやく知ることができます。
▶**英単語暗記トレーニング**・・・単語カードを再現します。出現の順番をランダム化することも可能で、より効果の高い訓練が行えます。
▶**辞書**・・・大学入試レベル約４０００語が予め登録されています。また、添付の辞書登録ユーティリティにより、簡単に辞書の登録が行えます。

好評発売中！X68000用ソフト

歴史に残る！日コン連ＳＯＦＴ初のシューティングゲーム！
Ｄ－ＲＥＴＵＲＮ
原作・開発者　神戸大学情報統計部　赤坂賢洋　　¥５９８０
将来の家宝に！在庫謹少の為お買い求めをお急ぎください！
第３回ハイスコアコンテスト結果　（上位ベスト５）
①1217万点　鈴木直孝（神奈川）②716万点　宮沢毅（長野）③656万点　斉藤晋一郎（宮城）④655万点　須田健太郎（大阪）⑤626万点　寺島真一（福島）

アドベンチャーゲームインタプリタ！
電脳作家Ｖｅｒ２．０
原作・開発者　神戸大学情報統計部　村尾元　　¥５９８０
プログラムが組めなくても、アドベンチャーゲームが作れます！
グラフィックツール、サンプルシナリオ付きです。
史上初、自作シナリオの商品化チャンス付きです。

電脳作家用グラフィック、ミュージックデータ集
電脳作家グラフィック＆ミュージックライブラリー集
制作者　神戸大学情報統計部　細見格・赤坂賢洋　　¥３９８０
Ｄ＿ＲＥＴＵＲＮゲームミュージックが自作ソフトのＢＧＭに使えます！

■**シナリオコンテスト応募作品集**
電脳作家Ｖｅｒ２．０対応シナリオディスク集①　　¥２９８０
ＥＶＩＬ　ＥＹＥ：作　三上潤一郎、帰らざる森　：作　川合一広
自作シナリオディスクを日コン連に送ると、ＰＣ－９８、ＦＭ－ＴＯＷＮＳ版にも移植して、もれなく商品化されます！

郵送品貼付切手には、オール記念切手使用！
日コン連ＳＯＦＴ通信販売のご案内
現金書留、郵便振替（大阪5-4873 日コン連企画株式会社）、為替、定額小為替で、希望商品名、対応機種名、数量明記の上、お申し込みください。（送料はサービス。）
このうち、現金書留、定額小為替でお申し込みの場合には、例えば5,980円の商品の場合には、端数を切上げ6,000円分お送りいただいて結構です。この際のおつり20円は、商品発送時に同額の記念切手でお返しいたします。

お知らせ！

日本コンピュータクラブ連盟加盟団体募集中！
加盟費・会費不要。毎月、全国本部広報紙「つうてんかく通信」無料送付。

●**日コン連では、以下のスタッフを求めています。**
・日コン連全国本部（難波）、関東本部（自由が丘）付けスタッフ
・日コン連パソコンウィルス研究所非常勤スタッフ
・1990年4月創刊のパソコン雑誌のライターおよびエディター
お問い合わせは、下記まで。
日コン連全国本部 06-644-6901（代）/日コン連関東本部 03-701-3298

●**日コン連ＳＯＦＴユーザー登録のお勧め**
日コン連ＳＯＦＴのユーザー登録をされますと、日コン連関連のものについては、様々な便宜がはかられます。日コン連ＳＯＦＴのコピー等をお持ちの方は、これを機会に本物をお買い求めいただくようお願いします。また、本物をお持ちの方は、ユーザー登録をされることをお勧めします。

登録ユーザー便宜①
日コン連主催のパソコン通信による大学受験相談で特別な便宜がはかられます。Ｊ＆Ｐ　ＨＯＴＬＩＮＥ　　　　　で、今年も大規模な大学受験相談が行なわれますが、例年ある、パソコン通信装置を持たない人からの志望大学の受験情報がほしいなどという要望に対しては、部室に電話のある大学サークルへ対するものを除いて、質問および回答できないとお断りしていました。今回、日コン連ＳＯＦＴにユーザー登録されている方に限り、返信用切手を同封されますと、日コン連全国本部が責任を持って、代理でパソコン通信で質問事項を送信、各大学サークルから得られた回答を郵送させていただきます。ユーザー登録済みソフト名とおよその登録ハガキ返送日をご記入の上、全国本部までお申し出ください。

登録ユーザー便宜②
日コン連パソコンウイルス研究所（大阪）では、日コン連ＳＯＦＴのユーザー登録されている方に限って、ご自分のマシンにかかったウイルスの検出、復元のご相談に応じます。また、日コン連パソコンウイルス研究所開発のワクチンについては、その配布を登録ユーザーの方に優先的に行います。

日コン連SOFT保証
お客様のご都合により、同一種の新しいディスクとの交換を希望される場合には、そのディスクと360円分の切手をお送りください。折り返し、新しいディスクをお送りさせていただきます。

●問い合せ・申し込み先

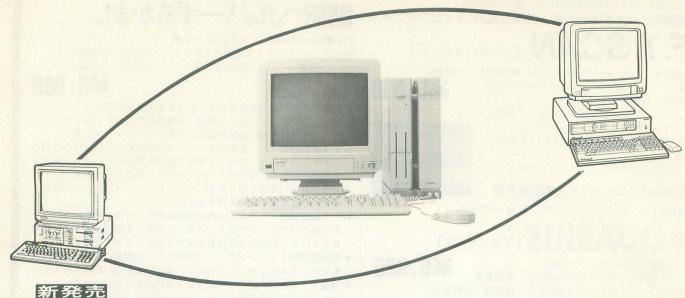
〒556　大阪市浪速区難波中2-4-3　村上ビル
TEL　06（644）6901（代）
日コン連企画株式会社または日本コンピュータクラブ連盟

外国製のMS-DOSにもアクセス出来る!

新発売

X68000用

SUPER DEVICE MONITOR "T"

X1turboや、MZ-2500ではもうお馴染の『SUPER DEVICE MONITOR "T"』のX68000用がいよいよ発売されました。

今までは、手探りで行なっていたプログラムの開発が、容易に出来る様に成ります。
例えばCコンパイラーや機械語を使ってソフトを自作している場合、1バイトの定数等を書き換えるのにいちいちエディターでソースプログラムを書き直してからアセンブルからもう一度やり直さなければ成らなかった作業が『SUPER DEVICE MONITOR "T"』を使うと1バイト単位で書き換えられるので簡単に出来る様に成ります。特にハンドアセンブルをする方には今までに無かった快適な開発環境を提供します。

★アクセスしたセクターは、縦横チェックサム付で表示して、ワープロ感覚で変更・複写・スクロール等の多彩なエディット機能が1バイト単位で使えます。

★S-RAMやIPLなど通常アクセス出来ない部分を含めてX68000内で呼び出せるメモリーは殆ど総てセクター単位でアクセス出来ます。

★RS-232Cを使うと任意のボーレートでX68000同士は勿論、他機種にはその機種用の『SUPER DEVICE MONITOR "T"』を介して、特殊なデータ圧縮法により、最高速では通常の32倍(理論値)の超高速で転送が行ええます。例えばフォーマットしたばかりの2Dのディスク1枚分を1200ボーで転送すると約8分間で転送が出来ます。
(X1のみ不可)

★256バイトを1セクターとしIPL-ROM、S-RAM、MIN-RAMなどが別々のディバイスとしてアクセス出来ます。

★X68000標準フォーマット以外のフォーマットもアクセス出来る可変フォーマット機能付です。

★RS-232Cのボーレートの変更は、ボタン1つで簡単に出来ます。

X68000用のみ最高1300ガウスの磁気を浴びても大事なフロッピーディスクが安全に守られる、三菱鉛筆製の磁気遮蔽機能付『uniフロッピーディスクケース』に入っています。

SUPER DEVICE MONITOR "T"

機種	サイズ	メディア	価格
X68000	5"	2HD	15,000円
X1	5"	2D	10,000円
X1turbo(2HDは受注生産)	5"	2D/2HD	13,000円
MZ-2500・MZ-2800	3.5"	2DD	13,000円

*MS-DOSはマイクロソフト社の商標です。
*商品の価格には消費税は含まれていません。

BLUE SKY Co.

▶お求めは全国の有名マイコンショップでどうぞ。
通信販売をご希望の方は当社へ直接、商品名・機種名・メディア名・住所氏名・電話番号を明記の上、現金書留にてお申し込みください。(送料無料)

株式会社 BLUE SKY
〒411 静岡県三島市加茂16-4
☎0559-72-6710

秋葉原価格でローンができます
電気の街秋葉原で
25年の信用!!

AVCフタバ

☎03(253)7661

〒101 東京都千代田区外神田3-2-3 ☎03-253-7611(代)

今すぐ もよりの電話から	仙 台 022-264-3704	名古屋 052-452-3271	広 島 082-295-6873
札 幌 011-611-5104	新 潟 0252-75-4175	大 阪 06-311-3931	福 岡 092-481-2494

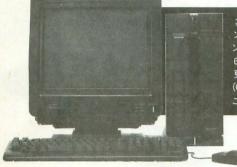
高価下取り、
買取りいたします、
お問合せ
下さい。

AVC本店 AVCジャンプ AVCホップ 至新宿 電波ビル 外堀通り リビナ 日通ビル 至上野 ヤマギワ 中央通り 至神田 秋葉原駅

Welcome!
ご来店もどうぞ。

X68000の情報のすべて!(当店はX68000の認定代理店です。お気軽にご相談下さい)

X68000 待望の新しい仲間登場!!

X68000 PERSONAL WORKSTATION EXPERT・EXPERT HD

集積度を高めた"マンハッタンシェイプ"2Mバイトのメインメモリを標準実装、Human68K ver2.0搭載(CZ-602C)更に40MBのHDDを搭載(CZ-612C)あくまでもX68Kにこだわるマシン。

〔写真のモニタは別売です。〕

CZ-602C 標準価格¥356,000
CZ-612C 標準価格¥466,000
AVC特価

X68000 PERSONAL WORKSTATION PRO・PRO HD

拡張I/Oスロットを4スロット標準装備、メインメモリ1MB、Human68K ver2.0搭載(CZ-652C)更に40MBのHDDを搭載(CZ-662C) 新しいX68Kの発見があるはずだ。

〔写真のモニタは別売です。〕

CZ-652C 標準価格¥298,000
CZ-662C 標準価格¥408,000
AVC特価

X68000 PERSONAL WORKSTATION ACE・ACE HD

従来機も忘れずに!!

CZ-611C-GY‥¥399,800
CZ-603D-GY·¥ 84,800
合計…………¥484,600
AVC特価
➡¥279,800

お勧めディスプレイコーナー 組合せは自由、価格はお気軽にご相談下さい。

CZ-612D 標準価格¥118,800 AVC特価
- 0.31mmドットピッチ
- TVチューナ搭載
- 3モードオートスキャン
- チルト台同梱

CZ-603D 標準価格¥84,800 AVC特価
- 0.31mmドットピッチ
- TVチューナ無し
- 3モードオートスキャン
- チルト台同梱

CZ-602D 標準価格¥99,800 AVC特価
- 0.39mmドットピッチ
- TVチューナ搭載
- 3モードオートスキャン
- チルト台同梱

CU-21CD 標準価格¥139,800 AVC特価
- 0.52mmドットピッチ
- TVチューナ無し
- 3モードオートスキャン
- チルト台取付不可

型 番	品 名	標準価格	販売価格
CZ-6TU	システムチューナー	¥ 33,100	AVCフタバ特価
BF-68PRO	CRTフィルター	¥ 19,800	AVCフタバ特価
CZ-8NS1	カラースキャナー	¥188,000	AVCフタバ特価
CZ-6BN1	スキャナー用パラレルボード	¥ 29,800	AVCフタバ特価
CZ-6VT1	カラーイメージユニット	¥ 69,800	AVCフタバ特価
CZ-8BV2	カラーイメージボード	¥ 39,800	AVCフタバ特価
CZ-8BR1	立体映像セット	¥ 29,800	AVCフタバ特価
CZ-8DT2	パーソナルテロッパ	¥ 44,800	AVCフタバ特価
CZ-8BS1	FM音源ボード	¥ 23,800	AVCフタバ特価
CZ-8NJ1	ジョイカード	¥ 1,700	AVCフタバ特価
CZ-8NM2A	マウス	¥ 6,800	AVCフタバ特価
CZ-8NM3	マウス・トラックボール	¥ 9,800	AVCフタバ特価
CZ-6SD1	システムラック	¥ 44,800	AVCフタバ特価
AN-S100	アンプ内蔵スピーカー	¥ 36,600	AVCフタバ特価
CZ-6EB1	拡張I/Oボックス	¥ 88,000	AVCフタバ特価

型 番	品 名	標準価格	販売価格
CZ-8PC3	24ドットカラープリンター	¥ 65,800	AVCフタバ特価
CZ-8PK7	24ピンプリンタ(80桁)	¥122,000	AVCフタバ特価
CZ-8PK8	24ピンプリンタ(136桁)	¥152,000	AVCフタバ特価
CZ-8PK9	24ピンプリンタ(80桁)	¥ 89,800	AVCフタバ特価
IO-735X	カラージェットプリンター	¥248,000	AVCフタバ特価
AP-800	48ドットカラープリンター(エプソン)	¥ 99,800	¥?9,000
VP-1000	24ピン(136桁)(エプソン)	¥154,000	¥?8,000
AP-550	24ドットカラープリンター(エプソン)	¥ 69,800	¥?9,000
CZ-6BE1A	1MB増設RAMボード	¥ 38,000	AVCフタバ特価
CZ-6BE2	2MB増設RAMボード	¥ 79,800	AVCフタバ特価
CZ-6BE4	4MB増設RAMボード	¥138,000	AVCフタバ特価
CZ-6BP1	数値演算プロセッサボード	¥ 79,800	AVCフタバ特価
CZ-6BC1	FAXボード	¥ 79,800	AVCフタバ特価
CZ-6BM1	MIDIボード	¥ 26,800	AVCフタバ特価
CZ-6BU1	ユニバーサルI/Oボード	¥ 39,800	AVCフタバ特価

型 番	品 名	標準価格	販売価格
CZ-8TM2	モデムユニット	¥ 49,800	AVCフタバ特価
CZ-252MS	Musicstudio	¥ 28,800	AVCフタバ特価
CZ-247MS	MUSIC(MID)	¥ 28,800	AVCフタバ特価
CZ-221HS	NEW Print Shop	¥ 19,800	AVCフタバ特価
CZ-228BS	TOP給与計算エキスパート	¥200,000	AVCフタバ特価
CZ-227BS	TOP財務会計	¥200,000	AVCフタバ特価
CZ-220BS	DATA	¥ 58,000	AVCフタバ特価
CZ-212BS	BUSINESS	¥ 68,000	AVCフタバ特価
CZ-219SS	OS-9	¥ 29,800	AVCフタバ特価
CZ-211LS	Ccompiler	¥ 39,800	AVCフタバ特価
CZ-234LS	AI-68K	¥188,000	AVCフタバ特価
CZ-620H	20MBハードディスク	¥178,000	AVCフタバ特価
CZ-64H	40MBハードディスク	¥120,000	AVCフタバ特価
LHD-34V	40MBハードディスク(ロジテック)	¥153,000	¥117,000
LHD-32V	20MBハードディスク(ロジテック)	¥128,000	¥ 98,000

CZ-8NJ2

アナログジョイステック
標準価格 ¥23,800

AVC特価¥???

X1turboZIII

X1ターボシリーズの独自の機能を全継承。VCCIゼロdB基準に適合させた。

CZ-888C…¥169,800
CZ-860D…¥ 99,800
合計………¥269,600
特価 ???

応談 価格はご相談に応じます。電話でお問い合せ下さい。

CZ-8PC4

48ドット熱転写プリンター。精密な文字、ハードコピーも可能。

CZ-8PC4……¥ 99,800

AVC特価¥???

IT X640

40MBハードディスク、OS-9、Human68Kの使用可。

アイテック
IT X640……¥158,000
特価 ¥118,000

特価¥118,000

●但し消費税(3%)は別途請求させていただきます。●分割回数は3回〜48回まで自由に選べます。

●セットの組合せは自由!広告に出ていない他の機種はお問合せ下さい。

●頭金なし(手軽な電話クレジット)●製品先取り(お支払いは約1〜2ケ月後から)●低金利クレジット(1回の支払いは2,700円以上上で3〜48回。ボーナス併用も可)●カレッジクレジット(保証人なし。但し満20歳以上の学生の方)●18歳未満の方(ご両親が代理購入者としてお申し込み下さい)●納期(通常の場合、当社に申込書が到着後1週間以内。特に人気のある商品で品薄の場合、少々納期が遅れることがありますので御了承下さい)●完全保証(すべてメーカー保証書付。アフターケア万全)●全国代引(お届けした者に、代金をお支払いいただく方法です。但し手数料1,000円)

AM10時からPM7時まで受付 日曜・祝日も営業

'90年もよろしくお願い致します

年末:年始:フェア

Macintosh

CALL

X68000 EXPERT	X68000 PRO	X1 turbo Z III
CZ602C CZ612D DiskCacher	CZ652C CZ103D DiskCacher	CZ888C CZ860D チルトスタンド
標準価格 ~~¥481,600~~-を BasicHouse特価	標準価格 ~~¥386,000~~-を BasicHouse特価	標準価格 ~~¥268,400~~-を BasicHouse特価

ディスプレイ

シャープ CZ602D	~~¥99,800~~
シャープ CZ612D	~~¥119,800~~
シャープ CZ603D	~~¥84,800~~

プリンタ

シャープ CZ8PC3	~~¥65,000~~	NEW CZ8PG2	¥?CALL	エプソン VP135EX	¥?CALL
シャープ CZ8PC4	~~¥99,800~~	NEW CZ8PK10	¥?CALL	エプソン VP1000	¥?CALL
NEW CZ8PG1	¥?CALL	シャープ CZ-6PV1	~~¥19,800~~	エプソン VP2000	¥?CALL

スキャナ

シャープ CZ8NS1	~~¥198,000~~	エプソン GT3000V	¥?CALL
エプソン GT4000	~~¥198,000~~	オムロン HS10R II	~~¥49,800~~
エプソン GT1000	¥?CALL	オムロン HS7R II	~~¥39,800~~

ハードディスク

シャープ CZ620H	~~¥88,000~~	アイテム HXD040	¥?CALL
シャープ CZ64H	~~¥120,000~~	アイテック ITX640	¥?CALL
ロジテック LHD34V	~~¥158,000~~	アイテック ITX680	¥?CALL

ジョイスティック

シャープ Cyber Stick	~~¥23,800~~
電波新聞社 XE1ST	¥?CALL
電波新聞社 XE1PRO	¥?CALL

拡張ボード

1M増設メモリ CZ6BE1A	~~¥38,000~~	増設RS232C CZ6BF1	~~¥49,800~~	コプロセッサボード CZ6BP1	~~¥79,800~~
2M増設メモリ CZ6BE2	~~¥79,800~~	MIDIボード CZ6BM1	~~¥26,800~~	CALL to BASIC HOUSE	
4M増設メモリ CZ6BE4	~~¥138,000~~	スキャナボード CZ6BN1	~~¥29,800~~		

モデム

アイワ PVA1200AIII	~~¥18,000~~	オムロン MD24FS5	~~¥49,800~~
オムロン MD1200AII	~~¥19,800~~	エプソン SR-240AT	¥?CALL

年末年始は BASIC HOUSE!

通信販売希望の方は、現金書留で(商品代金+送料¥1,000)×消費税1.03分を住所、氏名、希望商品名等を書いた紙を同封して御申し込み下さい。長期クレジットOK、支払い方法は御相談に応じます。

全国どこでも発送可　長期クレジットOK　送料全国均一¥1,000　宅配便にて即日配送

株式会社計測技研　本社営業部/マイコンショップ/通販部 〒321 宇都宮市竹林町503—1 TEL0286-22-9811 FAX0286-25-3970

マイコンショップ BASIC HOUSE　お申し込み・お問い合せは ☎0286-22-9811(代)

少ないスロットを上手に活かす

コプロセッサと増設メモリを1枚のボードに収納

SHARP純正品の2M/4M増設メモリと数値演算プロセッサボードとコンパチブル

新発売 KGB-X68PRK

実装メモリによって選べる4タイプ
購入後のメモリ増設も可能
コプロセッサ別売り(価格はお問い合わせ下さい)

標準価格

1MRAM	¥58,000
2MRAM	¥74,000
3MRAM	¥98,000
4MRAM	¥122,000

注　意）★初期型/ACE/PROの場合専用1Mバイト RAMを増設してメインメモリを2Mバイト以上にしている必要があります。
おわび）★発売が大変遅れましたことをおわび申し上げます。

64180CPU on X68000 Mach(マッハ)180

- CPUにHD64180(クロック10MHz/ノーウェイト)を採用、クラス最高速。
- メモリーは64kバイトを実装、67k CP/M相当(TPA 62kバイト)として使用可能。
- CP/M-80 BDOSエミュレータの使用によりBDOSレベルでのCP/M-80互換を実現。
- Human 68kのコマンドと同一ディスク上での混在使用が可能。
- モード切り替えの必要は一切なし。
- CP/Mディスクドライバによりturbo CP/M(2HD)のフロッピーの直接アクセスが可能。

Z80/HD64180のプログラムをX68000上で開発できる!

発売中
標準価格 ¥98,000

X68000 X1turbo オリジナルハードウェア

12bit 16ch A/D converter KGB-X68ADC
X68000 ¥128,000

12bit 16ch A/D converter KGB-AD12
X1turbo ¥118,000

12bit 4ch D/A converter KGB-DA4
X1turbo ¥98,000

高絶縁型16bit PIO KGB-X68ADC
X68000 ¥68,000

高絶縁型16bit PIO KGB-PIO(X1)
X1turbo ¥42,000

ローコスト汎用A/D & PIO ADC0809 & 8255PIO KGB-X1S
X1turbo ¥19,800

GP-IB INTERFACE KGB-488
X1turbo ¥58,000

HANDY PRINT Jack KGU-HDPR
X68000 ¥24,800

X68000 X1turbo オリジナルソフト&ハードウェア

X68000用ユニバーサル基板 KGB-X68UNB ¥6,800	X68000用MIDIインターフェース MELODYBOX ¥16,800	MZ2000用汎用A/D & PIO KGB-MZ1 ¥15,500	X1 turbo用HDDインターフェース KGB-HDIF ¥16,000
BASIC拡張関数パッケージ XBASICに約50種類の関数を付加 B6-6301 ¥9,800	C言語ライブラリ B6-6301をXBAStoCで使用出来る B6-6305 ¥6,800	BASIC拡張関数パッケージ C言語ライブラリ付き B6-6306 ¥14,800	Toys & Tools Human68kに約70個のコマンドを追加 B6-6307 ¥6,800
アイコンエディタ VSのアイコンを手軽に作成 B6-6303 ¥4,800	CP/M68Kエミュレータ Human上でCP/M68Kをエミュレート B6-6302 ¥19,800	ディスクキャッシャー FD/HDのアクセスの高速化 B6-6304 ¥6,800	BASIC HOUSE

全国どこでも発送可　長期クレジットOK　送料全国均一¥1,000　宅配便にて即日配送

株式会社計測技研　本社営業部／マイコンショップ／通販部　宇都宮市竹林町503—1　TEL0286-22-9811　FAX0286-25-3970

マイコンショップ BASIC HOUSE　お申し込み・お問い合せは ☎0286-22-9811(代)

アイ・ツー
大阪店
2F
EXECLUB
オープン
全国初
店頭大集合
新規ユーザー
EXE会員
X68000 EXPERTコーナー
★オープニングセール(12月1日〜1月31日)
期間中にご来店頂きました方にはステキなプレゼント進呈!!
ソフト・書籍コーナー
★X68000ユーザーニーズに対応したハード・ソフトウェア・周辺機器は全て展示しています。
★新製品情報・ユーザー同士の情報交換ができる、メンバー様の憩いのスペースです。
★期間中X68000・ディスプレイ・プリンター御購入の方は全国どこでも送料無料!!
遠くでなかなかお越し頂けない方にも通販専用TELで専門スタッフ(X68PRO STAFF)が親切丁寧にお答えします。
★X68000お買い上げの方はBIG・プレゼント!!
さらに大ヒットしているサイバースティック(CZ-8NJ2)プレゼント(早い者勝)
限定
X68000 goods
★頭金ナシ毎月¥3,000からあなたのプランにあわせてアイ・ツーラクラククレジット!
手数料もグーンと超低金利
X68000 PRO コーナー
★アイ・ツーメンバーズ優待制度実施
アイ・ツーでX68000・及びソフトウェア周辺機器をお買上げ頂きましたユーザー様にはオリジナルメンバーズカードを送付致します。メンバーズの方には楽しいパソコンライフをおくれますように最善のフォローをアイ・ツーより提供します。
御堂筋
南街劇場
高島屋
ナンバ
ナンバシティ
南海電鉄
アイ・ツー大阪店
ナンバメガネ
ニノミヤパソコンランド
地下鉄近鉄日本橋
地下鉄堺筋線
吉野屋牛丼
地下鉄恵美須町
■営業時間 AM11:00〜PM8:00
通販専用TEL.
06-634-0012
634-1198
年中無休
Information & Interface
株式会社アイ・ツー
大阪店 〒542 大阪市中央区難波千日前15-18 ☎06-632-0012代

JPDMA 安心と信頼のシステムで新時代を切り開く

Let's CALL 高額下取りセール実施中!!今すぐお電話下さい。03(797)1221

X68000

EXPARTシリーズ
・PROシリーズ新登場!!

・オリジナルOS「Human68k ver. 2.0」を搭載
・40MBハードディスクドライブを内蔵
・メインメモリ2MB標準装備(EXPERTシリーズ)
・拡張I/Oスロット4スロット内蔵(PROシリーズ)

☆注文No.A-0121

SHARP CZ-602C	¥356,000
SHARP CZ-602D	¥ 99,800
標準価格合計	¥455,800
現金特別価格	~~¥455,800~~

大特価にて提供中

☆注文No.A-0122

SHARP CZ-612C	¥466,000
SHARP CZ-602D	¥ 99,800
標準価格合計	¥565,800
現金特別価格	~~¥565,800~~

大特価にて提供中

☆注文No.A-0123

SHARP CZ-652C	¥298,000
SHARP CZ-602D	¥ 99,800
標準価格合計	¥397,800
現金特別価格	~~¥397,800~~

大特価にて提供中

☆注文No.A-0124

SHARP CZ-662C	¥408,000
SHARP CZ-602D	¥ 99,800
標準価格合計	¥507,800
現金特別価格	~~¥507,800~~

大特価にて提供中

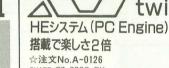
当社はX68000 PRO SHOPです。

●どこよりもお得な高額下取り実施中!! セットの組合わせは自由自在、ぜひご相談下さい。

X1 turbo Z III

画像取り込み、ビデオ編集、ステレオFM音源、多才な機能でひろがるアートワーク。

☆注文No.A-0125

SHARP CZ-888C-BK	¥169,800
SHARP CZ-860D-BK	¥ 92,200
標準価格合計	¥262,000
現金特別価格	~~¥262,000~~

大特価にて提供中

X1 twin

HEシステム(PC Engine)搭載で楽しさ2倍

☆注文No.A-0126

SHARP CZ-830C-BK	¥ 99,800
SHARP CZ-830D-BK	¥ 90,600
標準価格合計	¥190,400
現金特別価格	~~¥190,400~~

大特価にて提供中

●どこよりもお得な高額下取り実施中!! セットの組合わせは自由自在、ぜひご相談下さい。

☆注文No.B-0123
"24ドット熱転写カラー漢字プリンタ"
SHARP CZ-8PC3 ¥65,800
現金特別価格 ~~¥65,800~~
大特価にて提供中

お支払例
①¥10,000×6回〔ボーナス〕無し
②¥ 3,200×20回〔ボーナス〕無し

☆注文No.B-0125
"48ドット熱転写カラー漢字プリンタ"
SHARP CZ-8PC4 ¥99,800
現金特別価格 ~~¥99,800~~
大特価にて提供中

お支払例
①¥9,500×10回〔ボーナス〕無し
②¥3,000×36回〔ボーナス〕無し

☆注文No.B-0147
"24ピン136桁漢字プリンタ"
SHARP CZ-8PK8 ¥152,000
現金特別価格 ~~¥152,000~~
大特価にて提供中

お支払例
①¥ 6,400×24回〔ボーナス〕無し
②¥12,100×12回〔ボーナス〕無し

☆注文No.B-0132
"インテリジェントコントローラ"
SHARP CZ-8NJ2 ¥23,800
現金特別価格 ~~¥23,800~~
大特価にて提供中

お支払例
①¥3,300×24回〔ボーナス〕無し
②¥6,200×12回〔ボーナス〕無し

中古在庫リスト

SHARP

本体

CZ-812C(X-1F model 20)	¥139,800 →	¥ 26,000
CZ-822C(X-1G model 30)	¥118,000 →	¥ 28,000
CZ-870C(X-1 TURBOIII)	¥168,000 →	¥ 52,000
CZ-880C(X-1 Turbo Z)	¥218,000 →	¥ 62,000
CZ-611C(X68000ACEHD) 新品同様	¥399,800 →	¥238,000
MZ-2861	¥328,000 →	¥148,000

ディスプレイ

CU-14G(14"2000文字カラーディスプレイ)	¥ 49,800 →	¥ 18,000
14M-522C(14"4000文字デジタルカラーディスプレイ)	¥ 99,800 →	¥ 42,000
CU-14A4(14"4000文字アナデジカラーディスプレイ)	¥ 89,800 →	¥ 42,000
CU-14H1(14"4000文字デジタルカラーディスプレイ)	¥ 99,800 →	¥ 42,000
CU-14BD(14"カラー4050/2000文字)	¥ 64,800 →	¥ 40,000
CU-14CD(14"カラー4050/2000文字) 新品同様	¥ 84,800 →	¥ 52,800
CU-14FD(14"4000文字アナログカラーディスプレイ) 新品同様	¥ 74,800 →	¥ 51,000
MZ-1D22(14"4000文字MZ用カラーディスプレイ)	¥108,000 →	¥ 45,000
CZ-611D(15"3モードスキャンカラーディスプレイTV) 新品同様	¥134,000 →	¥ 82,000

SHARP
CZ-611CGY 新品同様
(X68000 ACE HD)
¥399,800→¥238,000
X68000 ACE HDディスプレイセット
(本体+CZ-611DGY) 新品同様
¥533,800→¥320,000

ディスクドライブ・プリンタ・他

CZ-8PC2(10"24ドット漢字プリンタ)	¥ 69,800 →	¥ 38,000
CZ-8PD2(80桁ドットプリンタ)	¥ 79,800 →	¥ 28,000
MZ-1P07(80桁漢字サーマルプリンタ)	¥ 79,800 →	¥ 22,000
CZ-8SS2(システムスタンド) 新品同様	¥ 5,500 →	¥ 4,000

その他各種在庫をとりそろえております。御気軽にお問い合せ下さい。

全商品保証付 中古も6ヶ月の保証期間だから安心です。	**クレジットでOK** カレッジクレジットも取扱います。
全国無料配送 お買上1万円以上、配達料はいただきません。	**日曜配達可** 留守の多い方でも安心です。
ショールーム Xシリーズ展示中。	**高額買取り** 電話1本で即、現金お支払い。
代金引換えシステム 商品到着時の代金支払いでOK。	**ボーナス一括払い** 商品は即お手元へ、お支払いはボーナス時に。

●電話一本で高額下取り、即商品はお手元へ!
●あなたの不要になったパソコンを電話一本で査定し買取ります。
●掲載の商品以外も取り扱っております。
●ビジネスソフトスクール受講者受付中!お気軽にお電話下さい。

▼本社注文デスク
03(797)1221
コンピュータバンク

株式会社パシフィックコンピュータバンク 〒150 東京都渋谷区渋谷1-6-8井上ビル 営業時間/平日AM9:30〜PM9:00 土・休日AM9:30〜PM8:00 (12/31 PM6:00迄 1/1〜3休業)

●クレジット価格に消費税は含まれておりますが、現金特別価格には含まれておりません 別途消費税かかかります

株式会社 デンキヤ

営業時間AM11:00～PM7:00 水曜定休

（価格は全べて税込みです）

セット超特価

X68000 PERSONAL WORKSTATION

PRO・PRO HD

CZ-652C	¥298,000
CZ-602D	¥99,800
定価合計	¥397,800
デンキヤ特価	¥287,800
CZ-662C	¥408,000
CZ-602D	¥99,800
定価合計	¥507,800
デンキヤ特価	¥367,000

セット超特価

X68000 PERSONAL WORKSTATION

EXPERT・EXPERT HD

CZ-602C	¥356,000
CZ-602D	¥99,800
定価合計	¥455,800
デンキヤ特価	¥329,800
CZ-612C	¥466,000
CZ-612D	¥119,800
定価合計	¥585,800
デンキヤ特価	¥423,000

全品メーカー保証 即決クレジットOK

ディスプレイ		プリンタ		周辺機器		ソフト	
CZ-603D	¥61,600	CZ-8PC3	¥51,400	CZ-8NJ1	¥1,400	CZ-213MS	¥15,500
CZ-602D	¥72,900	CZ-8PC4	¥77,250	CZ-8NJ2	¥18,540	CZ-223CS	¥15,300
CZ-612D	¥87,550	CZ-8PK8	¥116,400	CZ-6BEIA	¥29,400	CZ-219SS	¥23,100
CU-21CD	¥101,970	CZ-8PK9	¥70,100	CZ-6TV	¥72,000	CZ-211LS	¥30,800

24時間テレホンサービス

0482-54-3444

お申し込み

TEL.0482-54-3400

FAX.0482-54-3443

埼玉県川口市西川口4-6-4

お支払い

下記取引銀行口座までお振込み下さい。

三菱銀行西川口支店

㈱デンキヤ㊢0258081

冬だX68000の季節

おこた・やきいも・68K

ボーナス・
新春セール
実施中！

X68000シリーズ

EXPERT	定価~~¥356,000~~
EXPERT HD	定価~~¥466,000~~
PRO	定価~~¥298,000~~
PRO HD	定価~~¥408,000~~

各シリーズとも特価販売中！
T・ZONE 2Fにて。

ADO・TOYOMURA T・ZONE ティー・ゾーン

Micom Zone

2F 〒101 東京都千代田区外神田4-4-1 ☎257-2650

T・ZONEからのクリスマスプレゼント

冬の「エキサイティング・バザール」開催

中古品・展示品・現品処分品を大特価で大放出!!

日時：1989年12月23日(土)・24日(日)
午前9時30分～午後5時
場所：秋葉原「通運会館」2F特設会場
住所：〒101 東京都千代田区外神田3-16-18

X68000をトータルサポート
T・ZONE 2F

SHARP Authorized

X68000 PRO SHOP

マイクロウェアシステムズ設立に伴う OS9-68Kサポートの移動のお知らせ

この度、米マイクロウェアシステムズの日本法人であるマイクロウェアシステムズ設立に伴い、X68K用OS9のサポート業務が今までのマイクロウェアジャパンからマイクロウェアシステムズにもとに移ります。そのため、X68K用SRC-DbgおよびCSG-IMSの発売が延びることが予想されます。宜しくご了承下さいますようお願い申し上げます。

今月の特価 10台のみ！

高速熱転写カラープリンタ
CZ-8PC3 ¥CALL

電子手帳とパソコンなど

ステーショナリー
PRO68K
¥14,800

注：電子手帳との接続には別売ケーブルCE200L(¥2,500)が必要です。

OS-9/68000 for X68000

□OS9/68000 (SHARP) ¥29,800

□C＆PRO PACK(マイクロウェア) ¥58,000

□MW-BASIC(マイクロウェア) ¥60,000

□BTree09 (ARK) ¥36,000
MW-BASIC用のISAM用B-Treeパッケージです。応用例として住所録と販売管理プログラムが付属。全ソースコード付です。(このソフトを動かすためにはMW-BASICが必要です。)

□UD-CACHE (ARK) ¥16,000
すべてのRBFデバイスに対応するキャッシュです。

□FBU (ARK) ¥38,000
ハード・ディスクバックアップユーティリティーです。巨大ファイルを分割バックアップしたり、日付管理を行なったバックアップもOK。

□VSED (FORKS) ¥28,000
OS9/68000で唯一オートバッファリングをサポートしたスクリーンエディタです。

□OS9-68K V2.0テクニカルマニュアル
ちょっと高いんですが、システムコールの説明からパイプライン、イベント等の情報が得られる貴重な一冊です。 ¥18,500

番外 Oh! FMコーナー

FMシリーズ用OS9に新ソフト登場！

日本ソフトバンク
DB-09 (FM-7, 77, AV, 11) ¥18,252！
OS9上で走るリレーショナルデータベースマネージャーです。問い合わせ形式で取扱い簡単。なんとCによる全ソース付。

IMAGE and TEXT'S Inc.
Plot it！ (FM-11) ¥16,500
OS9上で走るプリント基板パターン設計用CAD。なんとVTerm 25にも対応。

星光電子感謝セール ～12月末まで

日頃のご愛顧にお応えして一部商品を除き(OS9News、定価1万円以下の商品)All 20%OFFでご提供します。

T-ZONE TZONE
8F オフィス
7F 工学書 輸入
6F 無線 衛星放送
5F HAM トランシーバー
4F トレード ゾーン
3F IBM ゾーン
2F パソコン ソフト ウェア
1F パーソナル OA
B1 Macintosh!

T・ZONE正社員・長期アルバイト募集中！
☆お問い合わせは総務課鈴木まで(TEL 03-257-2630)

T・ZONE
営業時間：AM10:30～PM7:00

下記T・ZONE各店でも扱っています。

宇都宮店：☎0286(63)4949 川口店：☎0482(68)7826 静岡店：☎0542(83)1331 横浜店：☎045(641)7741
大宮店：☎048(652)1831 東ラジ店：☎03(257)2694 パーツショップ：☎03(257)2655

●マイコン通販利用の方へ：現金書留て送金される際は、住所、氏名、TEL番号、希望商品名(詳しく)を明記して下さい 振込を御希望の方は下記銀行へお願いします
尚、いずれも予めTELにて、御予約・送料確認の上御送金下さい (振込口座 埼玉銀行 秋葉原支店 当座2705 株亜土電子工業)

☆この広告の提示価格には、消費税は含まれておりません。

安心と信頼の誌上ショッピング メディアショップ お申込みは今すぐ電話かハガキで!!

株式会社 メディアショップ ハイランド　〒239 神奈川県横須賀市ハイランド3-9-6

電話でのお申込みは

お申し込みはフリーダイヤルで(料金無料)

0120-483290

お問合せは専用ダイヤルで

0468-483290

年中無休AM10時〜PM10時

ハガキでのお申込みは

〒239
神奈川県横須賀市
ハイランド3-9-6
(株)メディアショップ
ハイランド
Oh!X 係

申込書
- 商品名(商品番号)
- 支払回数
- お名前
- 生年月日
- ご住所、電話番号
- お勤め先
  名称、住所、電話番号

通信販売のお申込み方法

▶現金一括でお申込みの方

- 商品名(商品番号)及び、住所、氏名、電話番号、ご覧の雑誌名をご記入の上、代金を現金書留でお送り下さい。
- 振込をご希望の方は、必ずお振込前にお電話又はおハガキで、お知らせ下さい。
  〈銀行振込〉協和銀行・久里浜支店　当座No.2945
  〈郵便振替〉横浜9-42177

▶クレジットでお申込みの方

- 電話かハガキでお申込み下さい。
  クレジット申し込み用紙をお送り致しますので、ご記入の上、当社へお送り下さい。

SHARP X68000 EXPERT

- CZ-602C(FDタイプ) 標準価格 356,000円
- CZ-612C(HDタイプ) 標準価格 466,000円
- CZ-602D(ディスプレイテレビ) 標準価格 99,800円
- CZ-612D(ディスプレイテレビ) 標準価格 119,800円
- CZ-603D(ディスプレイ) 標準価格 84,800円

SHARP X68000 PRO

- CZ-652C(FDタイプ) 標準価格 298,000円
- CZ-662C(HDタイプ) 標準価格 408,000円
- CZ-602D(ディスプレイテレビ) 標準価格 99,800円
- CZ-612D(ディスプレイテレビ) 標準価格 119,800円
- CZ-603D(ディスプレイ) 標準価格 84,800円

X68000 超特価セール!!

セットの組合わせも自由自在です。
まずはお問合わせ下さい。

EXPERT グラフィックス

- CZ-612C(本体)……466,000円
- CZ-612D(ディスプレイテレビ)……119,800円
- CZ-8NS1(イメージスキャナー)……188,000円
- CZ-6BN1(パラレルボード)……29,800円
- CZ-8PC4(48ドットカラープリンタ)……99,800円
- A-400HP(ビデオデッキ)……104,800円
- CZ-221HS(NEW Print SHOP)……19,800円
- CZ-235GS(グラフィックライブラリV.1)…8,800円
- CZ-236GS(グラフィックライブラリV.2)…8,800円

標準価格 1,045,600円

商品番号 227	一括払価格 814,000円
初回15,348円・12,500円×47回	ボーナス50,000円×8回
初回12,860円・10,800円×59回	ボーナス40,000円×10回

EXPERT 通信・パソコンFAX

- CZ-612C(本体)……466,000円
- CZ-603D(ディスプレイ)……84,800円
- BF-68PRO(CRTフィルター)……19,800円
- CZ-8TM2(モデムユニット)……49,800円
- CZ-8PK8(24ピン漢字プリンタ)……152,000円
- CZ-6BC1(FAXボード)……79,800円
- CZ-223CS(Communication)……19,800円

標準価格 872,000円

商品番号 219	一括払価格 694,000円
初回13,308円・11,100円×47回	ボーナス40,000円×8回
初回11,160円・9,900円×59回	ボーナス30,000円×10回

PRO データベース

- CZ-662C(本体)……408,000円
- CZ-612D(ディスプレイテレビ)……119,800円
- CZ-8PC4(48ドットカラープリンタ)……99,800円
- CZ-8NS1(イメージスキャナー)……188,000円
- CZ-6BN1(パラレルボード)……29,800円
- CZ-220BS(DATA PRO68K)……58,000円
- CZ-226BS(CARD PRO68K)……29,800円

標準価格 933,200円

商品番号 229	一括払価格 757,000円
初回13,324円・11,900円×47回	ボーナス45,000円×8回
初回12,930円・10,400円×59回	ボーナス35,000円×10回

EXPERT サウンド【MIDI】

- CZ-602C(本体)……356,000円
- CZ-602D(ディスプレイテレビ)……99,800円
- AN-S100 (アンプ内蔵スピーカーシステム) 36,600円
- CZ-6BM1(MIDIボード)……26,800円
- CM-32L(MIDI音源モジュール)……69,000円
- CZ-252MS(Musicstudio V1.1)……28,800円
- CZ-248MS(ソングライブラリ)……8,800円
- CZ-247MS(MUSICPRO68K MIDI)…28,800円

標準価格 654,600円

商品番号 228	一括払価格 542,000円
初回 9,444円・8,900円×47回	ボーナス30,000円×8回
初回 9,480円・8,300円×59回	ボーナス20,000円×10回

PRO ワープロ

- CZ-652C(本体)……298,000円
- CZ-603D(ディスプレイ)……84,800円
- BF-68PRO(CRTフィルター)……19,800円
- CZ-8PK8(24ピン漢字プリンタ)……152,000円
- EW(日本語ワープロソフト)……38,000円

標準価格 592,600円

商品番号 221	一括払価格 477,000円
初回 9,264円・7,200円×47回	ボーナス30,000円×8回
初回 8,230円・6,900円×59回	ボーナス20,000円×10回

X1, X1turbo X68000シリーズ用周辺機器

カラービデオプリンタ

- CZ-6PV1
  パソコンやビデオ機器に対応。64階調(485×480ドット)で再現する、昇華性染料熱転写方式を採用。

標準価格 198,000円

商品番号 149	一括払価格 163,000円
24回	初回 8,720円・7,700円×23回
36回	初回 5,862円・5,300円×35回

カラー イメージ スキャナー

- CZ-8NS1
  高速、高精度でハイレベルな画像入力を実現　最大A4サイズの原稿をフルカラー読み取り可能

標準価格 188,000円

商品番号 188	一括払価格 155,000円
24回	初回 8,800円・7,300円×23回
36回	初回 6,970円・5,000円×35回

48ドット熱転写カラー漢字プリンタ

- CZ-8PC4
  精緻で略字のない高品位印字。美文書もアートワークも鮮やかに、美しさの48ドットカラープリンタ

標準価格 99,800円

商品番号 216	一括払価格 82,000円
12回	初回 7,440円・7,300円×11回
24回	初回 6,080円・3,800円×23回

カラー イメージジェット

- IO-735X
  はがきからOHPフィルム、B4横サイズ対応。鮮明カラープリンタ。バッファメモリ(128KB)搭載。
- IO-73CX
  信号ケーブル。

標準価格 253,500円

商品番号 232	一括払価格 210,000円
36回	初回 8,540円・6,800円×35回
48回	初回 9,620円・5,300円×47回

商品名	型式	標準価格	販売価格
14型カラーディスプレイ	CZ-603D	84,800	71,900
RGBシステムチューナー	CZ-6TU	33,100	29,300
CRTフィルター	BF-68PRO	19,800	16,300
熱転写カラープリンタ	CZ-8PC3	65,800	54,000
漢字プリンタ(80桁)	CZ-8PK9	89,800	72,700
漢字プリンタ(80桁)	CZ-8PK7	122,000	98,400
漢字プリンタ(136桁)	CZ-8PK8	152,000	122,800
ハードディスク(20MB)	CZ-620H	178,000	143,700
増設用HDD(40MB)	CZ-64H	120,000	100,200
モデムユニット	CZ-8TM2	49,800	42,100

商品名	型式	標準価格	販売価格
カラーイメージユニット	CZ-6VT1	69,800	59,000
スキャナ用パラレルボード	CZ-6BN1	29,800	25,200
1MB増設RAM	CZ-6BE1	35,000	29,500
1MB増設RAM	CZ-6BE1A	38,000	32,100
2MB増設RAM	CZ-6BE2	79,800	67,300
4MB増設RAM	CZ-6BE4	138,000	116,400
ユニバーサルI/Oボード	CZ-6BU1	39,800	33,600
GP-IBボード	CZ-6BG1	59,800	50,400
増設用RS-232Cボード	CZ-6BF1	49,800	42,000
数値演算ボード	CZ-6BP1	79,800	67,300

商品名	型式	標準価格	販売価格
FAXボード	CZ-6BC1	79,800	67,300
MIDIボード	CZ-6BM1	26,800	23,200
拡張I/Oボックス	CZ-6EB1	88,000	74,200
システムラック	CZ-6SD1	44,800	37,800
スピーカーシステム	AN-S100	36,600	30,900
カラーイメージボードII	CZ-8BV2	39,800	33,500
立体映像セット	CZ-8BR1	29,800	24,600
インテリジェントコントローラ	CZ-8NJ2	44,800	20,100
FM音源ボード	CZ-8BS1	23,800	20,100
フロッピーディスクユニット	CZ-503F	49,800	39,200

商品名	型式	標準価格	販売価格
DATA PRO68K	CZ-220BS	58,000	49,300
CARD PRO68K	CZ-226BS	29,800	25,400
Sampling PRO68K	CZ-215MS	17,800	15,300
NEW Print SHOP	CZ-221HS	19,800	16,400
Communication	CZ-223CS	19,800	16,900
C compiler	CZ-211LS	39,800	34,500
Musicstudio V1.1	CZ-252MS	28,800	24,600
MUSIC(MIDI)	CZ-247MS	28,800	24,600
OS-9/X68000	CZ-219SS	29,800	25,400
Stationery	CZ-240BS	14,800	13,700

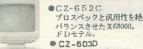
今月の特選お買得品(限定)

SHARP X68000 EXPERT

- CZ-602C
  ノーブルな機能美が、クリエイターを魅了する。妥協を排したこだわりのX68000。
  FDモデル。
- CZ-602D
  15型カラーディスプレイテレビ

標準価格 455,800円

商品番号 223	一括払価格 359,000円
48回	初回9,888円・9,200円×47回
60回	初回8,810円・7,700円×59回

SHARP X68000 PRO

- CZ-652C
  プロスペックと汎用性を絶妙にバランスさせたX68000。
  FDモデル。
- CZ-603D
  14型カラーディスプレイ

標準価格 382,800円

商品番号 212	一括払価格 304,000円
48回	初回7,928円・7,800円×47回
60回	初回8,660円・6,500円×59回

安心と信頼 メディアショップ ハイランド

①完全保証 全国どこでもアフターケアOK
②全国無料配送 日曜配送可能
③支払回数は 予算に応じ3〜60回 ボーナス併用可
④消費税 広告は全て消費税込みの価格で表示してあります
⑤FAXでも注文OK FAX：0468(48)3273
⑥その他広告以外の商品も取扱っております。お気軽にお問合せ下さい。

SHARP X68000 EXEショップ

年末年始超特価セットSale

（このセットに限り、送料＋消費税込）

AIBIT
アイビット電子株式会社

X68000 EXPERT
PERSONAL WORKSTATION

《シャープ見体験フェア展示品》

本体＋ディスプレイ

CZ-602CBK（本体）　定価￥356,000
CZ-602DBK（ディスプレイ）　定価￥ 99,800
定価合計￥455,800

'90年1月末迄

セット超特価 ￥350,000

CZ-652C＋603Dセット（超特価￥300,000）も販売中!

※代金は商品引換着払いでもOKです。

こんな商品お探しではありませんか？AIBITならあるかも知れません。
旧型商品在庫処分価格はお問い合わせ下さい。

◎シャープ／BF12M／CE158／CZ117SF／CZ134SF／CZ8BE1／CZ8BGR2／MZ8BD03／MZ1C14／MZ1C17／MZ1C25／MZ1C28／MZ1C33／MZ1C37／MZ1E08／MZ1E14／MZ1E29／MZ1E30／MZ 1M08／MZ1R01／MZ1R11／MZ1R14／MZ1R16／MZ1R24／MZ1T03／M Z1U03／MZ1X26／MZ1X30／MZ1Z003／MZ1Z005／MZ1Z015／MZ2Z017／MZ2Z023／MZ2Z025／MZ2Z025A／MZ2Z027／MZ2Z028／MZ2Z0 31／MZ2Z032／MZ4Z001／MZ5Z013／MZ6P09／MZ6P13／MZ80DPK／ MZ80F05／MZ80F15／MZ80FIO／MZ80FMD／MZ80MCA／MZ80T20A ／MZ80T40A／MZ80T70A／MZ80T70B／MZ80TU／MZ80TUB／MZ8BC 03／MZ8BC04／MZ8BFC／MZ8BI04／MZ8BK／MZ8BT04／MZ8BT05／MZ8KP4C／MZ8KP4I／MZLOGO／WD05HS／WD580／その他
◎富士通／B273A040／B273D010／B273D020／B273D040／B273D050 ／B273D051／B278A100／B278E010／FM77-211／FM77-411／FM77-412 ／FM77-421／FM77EM256／FM77EM64／FM77FD302／FP1030／HIMEJI BAISC／MB22020／MB22021／MB22204／MB22407／MB22410／MB2 2414A／MB22414B／MB22415A／MB22436／MB22441／MB22501／M B22504／MB22505／MB22601／MB22602／MB26208／MB27150／MB 27407／MB27412／MB27413／MB27702／MB28021J／MB28122／SM 07217J011／SM07317D023／SM07317M024／SM07317M143／SM074 17F224／SM07418C819／SM07418M054／SM07818M056／SM 07818 M224／SM07A17M342
◎NEC／PC6011／PC6601-01／PC8044K／PC8801-07／PC8801SR-FD1 ／PC8821-02／PCKD902／PCPR405-01／NM9700／PC8023C／PCKD 301
◎日立／MP1802A／MP1806／MP9710／MP9740
◎IOデータ／EM6010／PIO2025-8／PIO2025A／PIO2029／PIO2030／ PIO2030ROM／PIO2032／PIO3022／PIO3024／PIO3025／PIO3030／ PIO3030ROM／PIO3039／PIO3040／PIO3050／PIO 9024C-2／PIO98LT
◎エプソン＃9050

FM TOWNS 新機種発売記念／下取り特価セール中

ハガキもOK、NewMZプリンタ
漢字カラー熱転写プリンタ
シャープMZ-1P22

標準価格￥59,800⇨
特価￥38,640（ケーブル付）
〈24×24ドット漢字・7色カラー・漢字30字/秒高速印字・MZ1P17とフルコンパチ・5KBのバッファメモリ付〉
適応パソコン→MZ2000、2500、5500、6500シリーズ、X1シリーズ、X68000シリーズ他。

"プリンタ・コピー・ファクス" 1台3役のスグレモノ
パソコンファクス **MZ-1V01**

●MZ25セット（インターフェース ソフト付）
標準価格合計￥342,800⇨**￥168,000**
●PC98セット（インターフェース ソフト付）
標準価格合計￥377,800⇨**￥198,000**
●MZ-1V01本体のみ
標準価格　￥278,000⇨**￥120,000**
※上記セットをご注文の際は3.5か5インチのご指定をしてください

シャープMZ-1X30モデムホン
（1×19上位機種）
〈300/1200BPS全2重通信対応モデム内蔵●音声入出力端子付●ダイヤルパルス/プッシュボタン対応●プッシュボタン音解析機能●シャープ手順、CCITT、V25bis通信手順サポート〉
標準価格￥98,000⇨**特価￥39,800**

東芝 J3100SS＋クレオ BUSI COMPO セット
J3100SS
定価￥198,000
BUSI COMPO
定価￥40,000
定価合計￥238,000
セット大特価￥195,000

アイビット推奨ディスプレイ

●シャープMZ-1D27
（アナログ/デジタル）
ドットピッチ 0.31
定価￥12,000⇨
特価￥83,000
MZ-1D27対応パソコン機種：MZ-2500・MZ-2861・MZ-6500・MZ-2000/2200・MZ-700/1500・CXシリーズ・PCシリーズ
（色はグレーのみ）

●シャープCZ-830D・BK
（14型）
2モードオートスキャン方式
（アナログ/デジタル）
定価￥98,000⇨
特価￥54,800
CZ-830D対応パソコン機種：CZ880C/881C。XI/TURBOシリーズ。ケーブルは本体付属を使用。NEC PC-8801・9801シリーズ（XA・XLのみ不可）MZ700/1500/2000/2200/2500各シリーズ（推奨品シャープ8D8K）。

●シャープCZ-611D-GY
（15型アナログTV/3モードオートスキャン）
定価￥145,000⇨
特価 ￥89,800
CZ-611D対応パソコン機種：※X1シリーズ/※X1 turboシリーズ/X1 yurboZシリーズ/X68000シリーズ/PC8801シリーズ/PC-9801シリーズ/PC-286シリーズ
（※は接続ケーブルAN1506が必要です）

●シャープCZ-602D-GY・BK
（15型カラーディスプレイTV）
ドットピッチ3.9
定価￥99,800⇨
特価￥89,000
CZ-602D対応パソコン機種：※X1シリーズ/※X1 turboシリーズ/X1 yurboZシリーズ/X68000シリーズ/PC8801シリーズ/PC-9801シリーズ/PC-286シリーズ
（※は接続ケーブルAN1506が必要です）

本 体／新旧在庫機種（新品）
●シャープ/CZ-602C/CZ-612C/CZ-652C/CZ-662C/CZ-801C/CZ-802C/CZ-803C/CZ-804C/CZ-820C/CZ-822C/CZ-888C/MZ-2200/MZ-2861/MZ-3500/MZ-5511/MZ-6556
●富士通/FM-NEW7/FM77AV/FM77AV1/FM77AV2/FM77AV20/FM77AV40/FM77D2/FM77L2/TOWNS1/TOWNS2
●東芝/J-3100SL/J-3100SS
●NEC/PC9801CV21/PC9801E/PC9801LV21/PC9801RA2/PC9801RX2/PC9801UV21/PC9801VX4/PC9801XA2

拡張機器他
●シャープCZ-8GR(X1.GRAM)‥￥32,000⇨￥12,000
●シャープCZ-8EP(I/Oポート)‥‥￥11,800⇨￥9,000
●シャープCZ-8EB3(I/Oボックス)‥￥33,800⇨￥28,000
●シャープCZ-8BK3‥‥(X1)‥‥￥13,800⇨￥11,700
●シャープCZ-8BK4‥‥(X1)‥‥‥￥6,800⇨￥5,700
●シャープCZ-8BGR2・(X1)‥‥￥14,800⇨￥4,000
●シャープCZ-8BS1‥‥(X1)‥‥￥23,800⇨￥19,500
●シャープCZ-64H(ハードディスク)〈CZ-602・652内蔵用〉‥‥￥120,000
●シャープCZ-8NJ2(インテリジェントコントローラー)￥23,800⇨大特価
●シャープCZ-8SS2システムスタンド‥‥￥5,500⇨￥2,500
●シャープCZ-81Tチルトスタンド‥‥‥￥8,500⇨￥1,000
●シャープCZ-8RL1(データーレコーダー)‥￥24,800⇨￥16,000
●シャープMZ-1U08(1500,700拡張)‥￥25,000⇨￥12,000
●シャープMZ-1U03(1500,700拡張)‥￥35,000⇨￥15,000
●シャープMZ-1X22モデムユニット‥‥￥21,800⇨￥13,000
●シャープMZ-1R12 RAM‥‥‥‥￥35,000⇨￥8,000
●シャープMZ-1E29 (MZ)‥‥‥￥17,800⇨￥9,800
●シャープMZ-1E30(Z-2500ハードディスクインターフェイス)‥‥‥‥￥30,000⇨￥27,000
●シャープMZ-1U09‥(2500)‥‥￥9,000⇨￥7,200
●シャープMZ-1M03‥(5500)・￥69,000⇨￥35,000
●シャープMZ8BC04‥(2000)‥￥18,000⇨￥8,000
●シャープMZ-8B104‥(2000)・￥45,000⇨￥18,000
●シャープMZ-1R11‥‥(5500)・￥80,000⇨￥40,000
●シャープMZ-1R24‥(2500)‥￥22,000⇨￥6,000
●シャープMZ-1R26A‥(2500)・￥13,000⇨￥12,800
●シャープMZ-1R27A‥(2500)・￥13,000⇨￥10,000
●シャープMZ-1R28A・(2500) ￥13,000⇨￥10,000
●シャープMZ-1R29A‥(2500)・￥32,000⇨￥10,000
●シャープMZ-1T02‥(2000)‥￥19,800⇨￥8,500
●シャープMZ-1T03‥(1500)‥￥12,000⇨￥8,500
●シャープMZ-1X29‥‥‥‥‥‥￥13,800⇨￥11,000
●シャープMZ1R35(1MB増設メモリーボード)‥￥55,000⇨￥19,000
●シャープMZ1R36(1R35用増設1MBボード)￥45,000⇨￥15,000
●シャープMZ1E26(ボイスコミュニケーション)‥‥￥24,800⇨￥13,000
●シャープMZ-3500キーボード‥‥‥‥‥‥￥10,000
●シャープMZ-5500キーボード‥‥‥‥‥‥￥10,000
●シャープ2000/2200キーボード‥‥‥‥‥￥10,000
●シャープSS-SC28M(シンディーコピーキット)￥49,800⇨￥10,000
●シャープ1E35(ADPCMボード)‥‥‥‥￥49,800⇨￥13,000
●シャープ1E39(RE232C 2CHボード)‥‥￥39,800⇨￥13,000
●シャープX1、MZ用マウス‥‥‥‥‥‥特価￥4,800
●シャープX1用ジョイカード‥‥‥‥‥‥‥‥￥1,500
●富士通16βキーボード‥‥‥‥￥25,000⇨￥20,000

プリンター
●シャープCZ-8PK7(漢字プリンター)・￥122,000⇨￥97,600
●シャープCZ-8PK8(漢字プリンター)・￥152,000⇨大特価
●シャープCZ-8PK9(漢字プリンター)‥‥￥89,800⇨￥71,800
●シャープCZ-81P(801用プロッタプリンタ)‥‥‥‥￥1,000
●シャープCZ-8PC3‥‥‥‥‥‥￥65,800⇨￥49,800
●シャープCZ-8PC4(黒・グレー)・￥99,800⇨大特価
●シャープMZ-1P27‥‥‥‥‥￥268,000⇨￥214,400
●シャープMZ-1P28‥‥‥‥‥￥148,000⇨￥118,400
●シャープMZ-1P29‥‥‥‥‥￥168,000⇨￥134,400
●シャープ6P-11(カットシードヒート)‥‥￥95,000⇨￥35,000
●富士通FMPR-351‥‥‥‥‥￥250,000⇨￥125,100
●富士通FMPR-353‥‥‥‥‥￥198,000⇨￥115,000
●富士通MB-27413‥‥‥‥‥‥￥90,000⇨￥25,000
●富士通FMPR-201R1 (二水ROM) ￥23,000⇨￥11,000
●富士通MB27407(熱転写プリンタ)‥‥￥79,800⇨￥33,000
●NEC-NM9700(漢字カラー)・￥163,000⇨￥65,000

ディスプレー
●富士通FMTV-211(200)‥‥￥185,000⇨￥89,000
●富士通FMTV-152(200)‥‥￥109,000⇨￥58,000
●NEC PC-KD854(400)‥‥‥‥￥89,800⇨￥58,000
●シャープMZ-1D10(400-白黒)￥41,800⇨￥25,000

フロッピーディスク
●シャープCZ-503F‥‥‥‥‥‥￥49,800⇨￥30,000
●シャープCZ-502F‥‥‥‥‥‥￥99,800⇨￥60,000
●シャープCZ-520F‥‥‥‥‥￥118,000⇨￥70,000
●シャープCZ-53F‥‥‥‥‥‥‥￥19,800⇨￥9,800
●シャープCZ-300F(CZ-3PCM付)‥‥‥‥‥￥13,000
●シャープCZ-52F(F増設ドライブ代品)‥‥‥￥15,000
●シャープCZ-51F(ターボ増設ドライブ代品)‥‥￥15,000

ソフト
●春望クリエイティブII・(2500)・￥34,800⇨￥29,000
●ビジレス‥‥‥‥‥‥(2500)・￥48,000⇨￥42,000
●Hu-CAL日本語‥‥‥(2500)・￥45,000⇨￥30,000
●ぷりんとしょっぷ‥‥(2500)‥‥￥9,800⇨￥5,000
●FILE UTILITY UT-25F・(2500)‥‥￥6,800⇨￥6,000
●パーソナルCP/M6Z001(2500)‥‥‥‥‥‥￥14,200
●V2BASIC6Z010‥‥‥(2500)‥￥10,000⇨￥8,500
●FORTRAN(1P1213)‥(2500)・￥13,800⇨￥11,700
●C MZ2500 1P1214‥(2500)・￥13,800⇨￥11,700
●COBOL 1P1215‥‥(2500)・￥13,800⇨￥11,700
●C CZ116LF(X1)‥‥‥‥‥‥￥13,800⇨￥11,700
●COBOL CZ118LF‥(X1)‥‥￥13,800⇨￥11,700
●ランゲージマスターCZ128SF‥‥￥9,800⇨￥8,500
●シャープCZ-130F(ターボCP/M)‥‥‥￥14,800⇨￥12,500
●シャープCZ-133SF(モデムターミナル)‥￥25,000⇨￥6,500
●シャープX1・3インチCP/M‥‥‥￥16,800⇨￥5,000
●HUMAN68K CZ-244SS‥‥‥‥￥9,800⇨￥8,500
●シャープ1P-1251(MZ2861用)‥‥￥88,000⇨￥20,000
●シャープ1P-1252(MZ2861用)‥￥55,000⇨￥20,000
●シャープ1P-1253(MZ2861用)‥￥77,000⇨￥20,000
●シャープ1P-1254(MZ2861用)‥￥66,000⇨￥20,000

X68000関係ソフト
●マイクロソフトウェアージャパン「C&プロフェッショナルパッケージ」‥‥‥‥‥‥￥58,000⇨￥49,800
●シャープOS-9/X68000‥‥‥‥￥29,800⇨￥25,300
●シャープCZ-211LS‥‥‥‥‥￥39,800⇨￥33,800
●シャープCZ-6BE1‥‥‥‥‥‥￥35,000⇨￥29,000
●シャープCZ-6BE1A‥‥‥‥‥￥38,000⇨￥32,000

シャープNEWパーソナルワークステーション
●PC/ATコンパチPC8041‥‥‥‥‥‥‥‥‥￥998,000
●AXディスクトップMZ8302A‥‥‥‥‥‥‥￥278,000
●AXディスクトップMZ8306A‥‥‥‥‥‥‥￥458,000

《全商品新品完全保証付》　■シャープポケコン全商品販売中。カタログ、特価表ご請求ください（〒72）。

☎0426-45-3001～3
FAX.0426-44-6002
●営業時間/10:00～19:00●電話受付/20:00迄可●定休日/日曜日(祭日営業)
SHARP SUPER XEX SHOP
アイビット電子株式会社　〒192 東京都八王子市北野町560-5

上記の広告商品はすべて店頭販売もしております。

全国通信販売 北海道から沖縄まで

★送料はご注文の際にお問い合わせ下さい。
★掲載の商品は、すべて新品、保証書付きです。
★掲載の商品は充分用意してありますが、ご注文の際は、在庫の確認の上、現金書留または、銀行振込でお申し込み下さい。全商品クレジットでも扱っております。
★お申し込みの際は必ず電話番号を明記して下さい。
★商品、品切れの節はご容赦下さい。

富士銀行八王子支店　（普）1752505

●本誌発売時には、上記価格よりさらにお求めやすい価格に変更されている場合があります。●上記商品価格には消費税は含まれておりません。全ての商品に対し、別途3%の消費税金がかかりますのでご了承ください。

クリエイト特典

- ●全商品完全保証書付(メーカー保証)
- ●全国無料配達(一部離島の方は有料になります)
- ●配達日の指定OK(日曜・祭日にかかわらずお客様のご都合にあわせて配達します)
- ●どんな商品の組合せも自由自在(ご予算、用途に応じ自由自在にシステムアップできます)
- ●中古パソコン高額下取り(今お使いのパソコンをわずかな差額でグレードアップ)
- ●お支払い方法自由(低金利の均等払い、ボーナス一括払いもご利用ください)

営業時間(年中無休)
AM10:00〜PM7:00(日曜・祭日はPM6:00まで)

当社はX68000の販売認定店です。どんなことでも安心してご相談ください。

(セットでお買上げのお客様にお好きなゲームソフトとテレホンカードを差し上げます!!)

X68000 PRO

- ●CZ-652C(本体・キーボード・マウス)……¥298,000
- ●CZ-603D(カラー専用ディスプレイ)……¥ 84,800
- ●CZ-8N32(アナログジョイステック)……¥ 23,800
- ●ゲームソフト……¥サービス

■定価合計……¥406,600

クリエイト特価

均等払い	¥ 5,480×36回	¥ 3,970×48回	¥ 3,460×60回
ボーナス	¥30,000×6回	¥25,000× 8回	¥20,000×10回

X68000 PRO HD

- ●CZ-662C(本体・キーボード・マウス・40Mハードディスク)……¥408,000
- ●CZ-603D(カラー専用ディスプレイ)……¥ 84,800
- ●CZ-8PK1(15インチ漢字プリンタ)……¥ 97,800
- ●ブランクディスケット、用紙……¥サービス

■定価合計……¥590,600

クリエイト特価

均等払い	¥ 6,660×36回	¥ 4,700×48回	¥ 3,930×60回
ボーナス	¥35,000× 6回	¥30,000× 8回	¥25,000×10回

X68000 EXPERT

- ●CZ-602C(本体・キーボード・マウス)……¥356,000
- ●CZ-602D(カラーディスプレイテレビ)……¥ 99,800
- ●ゲームソフト、ブランクディスケット……¥サービス

■定価合計……¥455,800

クリエイト特価

均等払い	¥ 5,840×36回	¥ 4,440×48回	¥ 3,980×60回
ボーナス	¥25,000× 6回	¥20,000× 8回	¥15,000×10回

X68000 EXPERT HD

- ●CZ-612C(本体・キーボード・マウス・40Mハードディスク)……¥466,000
- ●CZ-612D(カラーディスプレイテレビ)……¥119,800
- ●CZ-240BS(ステーショナリプロ)……¥ 14,800
- ●PA-7500(電子システム手帳)……¥ 22,000
- ●CE-200L(電子手帳接続ケーブル)……¥ 2,500

■定価合計……¥625,000

クリエイト特価

均等払い	¥ 7,500×36回	¥ 4,985×48回	¥ 4,720×60回
ボーナス	¥45,000× 6回	¥40,000× 8回	¥30,000×10回

※本広告に掲載の全商品の価格について消費税は含まれておりません。

★この表以外の組合せ、お支払い方法もご自由にできます。
★X1シリーズ用、X68000シリーズ用各社ハードディスク/プリンタ等の周辺機器を大特価にて販売しております。
電話にてお問合せください。

X68000シリーズ用 周辺機器・ソフトお買い得セール

型番	品名	定価
CZ-6VT1	カラーイメージユニット	¥ 69,800
CZ-8NS1	カラーイメージスキャナ	¥188,000
CZ-6BE1A	IMB増設RAMボード	¥ 38,000
CZ-6BE2	2MB増設RAMボード	¥ 79,800
CZ-6BE4	4MB増設RAMボード	¥138,000
CZ-8NM3	マウス・トラックボール	¥ 9,800
BF-68PRO	高性能CRTフィルター	¥ 19,800
CZ-6BP1	数値演算プロセッサ・ボード	¥ 79,800
CZ-8NT1	トラックボール	¥ 13,800
CZ-6BM1	MIDIボード	¥ 26,800
AN-S100	アンプ内蔵スピーカースシステム	¥ 36,600
CZ-8NJ2	アナログスティック	¥ 23,800
CZ-603D	ドットピッチ0.31mm14型高解像度	¥ 84,800
CZ-6TU	パソコンチューナ	¥ 33,100

ソフト名	品名	定価
MUSIC PRO	MIDI版	¥ 28,800
MUSIC PRO-68K	マウスを使った楽譜ワープロ	¥ 18,800
SOUND PRO-68K	サウンドエディタ	¥ 15,800
Sampling PRO-68K	AD PCMサンプリングエディタ	¥ 17,800
Musicstudio PRO-68K V.1.1	MIDIマルチレコーディングソフト	¥ 28,800
NEW Print Shop PRO-68K	ポップアートツール	¥ 19,800
Communication PRO-68K	高機能通信ソフト	¥ 19,800
OS-9/X68000	マルチタスクオペレーティングシステム	¥ 29,800
PRO-68K	サイバーノート	¥ 19,800
PRO-68K	ステーショナリー	¥ 14,800
DATA PRO-68K	コマンド型リレーショナルデータベース	¥ 58,000
CARD PRO-68K	カード型リレーショナルデータベース	¥ 29,800
Ccompiler PRO-68K	ソフト開発セット	¥ 39,800
Human 68K Ver2.0	開発ツールセット	¥ 9,800

▲上記以外ビジネスソフト、最新ゲームソフト豊富に在庫あります。※送料はご注文の際お問合せください。 ●超特価販売中!

●渋谷店

●横浜店

総合お問合せ先☎03-486-6541(代)

パソコン専門ショップ
ソフトクリエイト 渋谷/横浜

●渋谷店☎03-486-6541(代)
〒150:東京都渋谷区渋谷1-12-7 三和渋谷ビル
振込銀行:三井銀行 渋谷宮益坂支店(普)No.5000340

●横浜店☎045-314-4777(代)
〒221:横浜市神奈川区鶴屋町2-12-8 第1建設ビル
振込銀行:三和銀行 横浜駅前支店(普)No.310852

近日完成

価格未定

No. 4

プログラム オペレーティング システム

複数のアプリケーションプログラムをバッチ処理の手軽さとC言語なみの制御コマンドで記述されたコマンドに従い順に実行するまったく新しいタイプのインタプリタです。

初心者からマニアまで、1番幅広く使われているのがバッチプログラムではないでしょうか。そんなプログラムにC言語なみの制御コマンドを持たせました。

実行するプログラムを選び必要なパラメータを与えながら流れに従って組上げる。こんな手軽さでプログラムが作れるなら初心者にも、なにか作れますね。

音楽をスタートし、絵を表示、次にナレーションを語る。これでも1つのプログラムです。バッチ処理なら3行です。本システムでも、このファイルを実行せよという記号を先頭に、1つ加えるだけで同じ処理をします。

各パラメータは渡しやすい様に、文字列に関する処理は内部コマンドでサポートしてあります。

If, While, Do, For, Switch などのコマンドを1つ1つ覚えながら処理を拡張していけば立派なプログラムに成長します。

For 文がどのように動作するか、1ステップ毎のトレース機能で確認でき、初心者にもその動作が良く分かり、誰にでもプログラミングが楽しめるように考慮してあります。

好評発売中

X68000専用
多機能デジタルサウンドツール

DiSS-P
ディスピー

Digital Sound System

豊富な機能をギッシリツメて、7,800円で登場!!

新時代の録音・編集・再生システム登場!

X68000専用に開発・設計しそのハイスペックを継承し、持つ機能を最大限に活用した、新しい時代の幕開けにふさわしいディスピーの誕生です。

特長

●すべてのサウンドをそっくりデジタル録音
ディスピー独自の長時間録音はナレーションからミュージックにいたるまであらゆるニーズに対応

●波形編集でプロフェッショナルなサウンドクリエイト
波形を確認しながら簡単なマウス操作でオリジナルサウンドをワンタッチでアレンジ

●ワンタッチ再生やプログラム再生など多彩な再生機能
●X68000が自在にしゃべる、スピーチ機能
●新時代のメール、ボイスメールシステム
●データは自作プログラムにそのまま利用可能
●ハイスピードなデータ処理とグラフ表示
●誰でも楽しめる豊富な音声データ付属
●買ったその日から使えるイージーオペレーション
●X68000が再生できるすべてのデータの編集が可能

※この他機能満載、使い方いろいろ、実用性を意識した仕様です。お気軽にお問合せください。
※改良のため、内容の一部を予告なく変更することがあります。

(※写真は1M増設時です)

通信販売

画面に皆様のお名前をお入れしてお届けします。住所・氏名ふりがなを明記し7,800円を、現金書留・郵便振替・銀行振込の何れかで下記宛にお願いします。(税込み・送料サービス)
郵便振替　東京 8-404042　サザン エンタープライズ
銀行振込　三和銀行　荏原支店　当座 308061

サザン エンタープライズ
〒142 東京都品川区戸越5-12-17 TEL・FAX 03-787-3932

パソコンで **3D GRAPHICS** を作ってみたいという諸君!!

このたび **Z's TRIPHONY DIGITAL CRAFT** ジーズトリフォニーデジタルクラフト による 玉川美土里著 **誰でもできる 3次元グラフィックス** という

本 が発売されました。この本を読むと

こんな絵 や こんな絵 を簡単に作ることができます。

内容は

- Z's TRIPHONY CRAFT 入門
- 立体のインビトゥーイン
- フラクタル図形の出力
- 図形いりモデリング解説

など盛りだくさん!

それになんと デジタルクラフト隠し機能 **全公開** !! **B5版 200頁** で定価 **2,500円**。

全国の書店で **工学図書** より 好評発売中!!

お問合わせは アーマットまで TEL 045(902)9041

フリーランスは
通信業態。

僕はフリーランスのカメラマン。
J&P HOT LINEで
シャッターチャンスを見つけ出す。

ロケ地探し/BBS

バッチリ狙いの風景を探すなら、地元の人に聞くのがいちばん。

フリーランスのカメラマンはフットワークが問題だ。遠くへ足をのばすにしても、狙い通りの風景をどれだけ的確に見つけられるかが、写真の仕上がりを左右する。こういう時は、J&P HOT LINEのBBS。全国の仲間にピタリのポイントを聞き出せば、効率的というものなのだ。

撮影計画/データベース

ホテルの所在も現地のまつりも、J&P HOT LINEで下調べ。

さて、撮影旅行に出かける前に、J&P HOT LINEのホテル情報で、撮影地の宿のあたりをつけておこう。全国各地の有名ホテルが一覧できるから、便利この上なし。現地で時間があまった時のために、当日行われそうな地元の祭りも調べておける。うーん、ゆとりだね。

ポラ送り/X-MODEM

とりあえずの仕上がりをクライアントに見てもらう。再撮影はまた明日。

撮影は順調に進んだ。今日の写真のいくつかは、クライアントに気に入ってもらえそうだ。さっそく、試しに撮ったポラロイドをパソコンのデータに変換し、得意先のCUG内へX-MODEMで送信する。画像データもおくれるなんて、なんて便利なネットなんだろ。

打ち合わせ/電子メール

次は雑誌のグラビアページ ロケ先からメールで打ち合わせ。

フリーランスは、時間の活かし方が重要になる。次の仕事は、都内でモデルの撮影だ。編集部のEさんに、スタイリストについて連絡をする。いつも忙しいEさんには、好きな時間に見てもらえる電子メールは、なにより確実な連絡方法。ムダのないのがありがたい。

批評会/SIG

時にはアマチュア時代を思い出す。新鮮な感覚がよみがえる。

さて、仕事の時間が終わったら、趣味の世界へ没頭しよう。SIGは、J&P HOT LINEの中のもうひとつのネットワーク。テーマを絞った議論についつい夢中。仕事で使う車についてモーターネットでひとくさり弁舌をぶってしまった。ああ4WDが欲しい。

情報収集/OLT (チャット機能)

仕事のネタも雑談の中から。脈絡のなさがいいのです。

SIGに書きこみをしていたら、たまたま同時にアマチュアカメラマンのSくんがアクセスしていた。ちょうどいいやと彼をさそってOLT(オンライン・トーク)へ入りこむ。日本中の人たちと同時に複数でおしゃべりしあえるOLTは、なんといっても、パソコン通信のダイゴミですな。

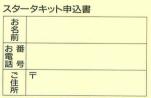
J&P HOT LINEは全国90ヵ所のアクセスポイント。2万人の仲間が、あなたの仲間になってくれます。

ご入会はスタータキットで
買ったその日からアクセスできます。

■申込書
〒556 大阪市浪速区日本橋5-6-7 上新電機株式会社
J&P HOT LINE事務局宛 TEL.(06)632-2521

■利用料金について
入会金/3,000円(スタータキット購入の代金から充当されます)
接続料/3分あたり20円(アクセスポイントまでの電話代は含みません)
※消費税3%が加算されます。

スタータキット申込書

お名前	
お電話番号	
ご住所	〒

お申込品 スタータキット(ソフトなし)
3,000+90(消費税3%)=¥3,090

●パソコン/ワープロ通信ネットワークサービス

J&P HOT LINE

アクセスポイントは全国に90ヵ所。日本全国を網羅する、本格的な通信ネットワークです。

スタータキットのお求めは、J&P各店でどうぞ。

店名	住所	電話
渋谷店	東京都渋谷区道玄坂2丁目28番4号	☎(03) 496-4141
町田店	東京都町田市森野1丁目39番16号	☎(0427)23-1313
八王子店	東京都八王子市旭町1番1号八王子そごう7F	☎(0426)26-4141
立川店	東京都立川市幸町4－39－1	☎(0425)36-4141
富山店	富山市双代町1番地	☎(0764)42-2131
金沢店	金沢市入江2－63	☎(0762)91-1130
大須店	名古屋市中央区大須4丁目2-48	☎(052)262-1141
テクノランド	大阪市浪速区日本橋5丁目6番7号	☎(06) 634-1211
メディアランド	大阪市浪速区日本橋5丁目8番26号	☎(06) 634-1511
コスモランド	大阪市浪速区難波中2丁目1番17号	☎(06) 634-3111
ワープロランド	大阪市浪速区日本橋4丁目9番15号	☎(06) 634-1411
ビジネスランド	大阪市北区梅田1-1-3大阪駅前第3ビルB2	☎(06) 348-1881
阪急三番街店	大阪市北区芝田1-1-3阪急三番街B1	☎(06) 374-3311
梅田店	大阪市北区小松原町1－10	☎(06) 362-1141
高槻店	高槻市高槻町11番16号	☎(0726)85-1212
くずは店	枚方市楠葉花園町15番2号	☎(0720)56-8181
千里中央店	豊中市新千里東町1-3-204千里サンタウン3F	☎(06) 834-4141
摂津富田店	高槻市大畑町24－10	☎(0726)93-7521
寝屋川店	寝屋川市緑町4－20	☎(0720)34-1166
藤井寺店	藤井寺市岡2丁目1番33号	☎(0729)38-2111
岸和田店	岸和田市土生町2451－3	☎(0724)37-1021
さんのみや1ばん館	神戸市中央区八幡通3-2-16	☎(078)231-2111
西宮店	兵庫県西宮市河原町5－11	☎(0798)71-1171
姫路店	姫路市東延末1丁目1番住友生命姫路南ビル1F	☎(0792)22-1221
京都寺町店	京都市下京区寺町通仏光寺下ル恵美須之町549	☎(075)341-3571
京都近鉄店	京都市下京区烏丸通七条下ル東塩小路町702	☎(075)341-5769
和歌山店	和歌山市元寺町4丁目4番地	☎(0734)28-1441
奈良1ばん館	奈良市三条町478－1	☎(0742)27-1111
郡山インター店	大和郡山市横田693－1	☎(07435)9-2221
熊本店	熊本市手取本町4－12	☎(096)359-7800

T4910217901563　雑誌02179-1